

Chat Modeling: Interaction-Enhanced Agent Framework for Visualizing Literature-Grounded Biological Structures

Donggang Jia[✉], Yunhai Wang[✉], and Ivan Viola[✉]

Abstract—Bioscientists frequently seek to visualize the biological systems they have empirically characterized and reported in the literature. Realizing such visualizations requires biological structure modeling, an inherently complex process that demands both biological and geometric understanding. This paper addresses the problem of constructing such 3D models for visualization. In this paper, we introduce a novel agent framework that mitigates the challenges of operating 3D modeling software by transforming user inputs, including natural language descriptions, research publication content, and textual descriptions of the existing objects and structures in the current scene, into modeling operations in a structured JSON format and final 3D results. The major technical contribution lies in the collaborative agent design that simultaneously supports model planning, execution, and novel user interaction design, such as interactive modeling execution and dynamic widget generation that fuse text and mouse interaction within the chat window. The framework further incorporates a customized modeling memory to enhance user interaction, featuring components such as personalized memory management, feedback collection, and skill library design. This modeling memory is leveraged to enable improved 3D modeling performance over time. The quantitative evaluation on our collected dataset showcases the effectiveness of our framework. We also develop a prototype tool, Chat Modeling, and demonstrate its usage through two modeling case studies. Our user study and expert interviews highlight the potential of our approach for use in scientific workflows.

Index Terms—Mesoscale modeling, molecular visualization, large language models

1 INTRODUCTION

Modeling and visualization of the cellular mesoscale, the scale bridging atoms and molecules with whole cells, is necessarily an integrative process, since no existing experimental method can directly observe cellular mesoscale structures [1]. In practice, biologists construct mesoscale models by synthesizing literature evidence and composing known biological components into faithful and visualizable 3D representations. However, transforming such evidence into manipulable 3D mesoscale models remains labor-intensive, requiring both biological expertise and proficiency with complex modeling tools. Current modeling tools, such as Maya [2]

and Blender [3], offer advanced modeling capabilities. However, these tools require users to interact with complex graphical user interfaces (GUIs) and possess a deep understanding of geometry and spatial relationships. This steep learning curve impedes users from effectively utilizing these tools, especially when modeling complex structures. For instance, structural biologists often struggle to translate their biological knowledge into the geometric and spatial constructs needed to create accurate 3D models [4]. We target the paper-to-model gap by enabling scientists to turn structures described in their papers into faithful, visualizable 3D models with minimal effort. Our agent framework takes text commands, scientific publications, and the current scene context as input, then plans and executes the modeling steps through a natural-language interface, thereby obviating the need for manual specification of complex geometric relationships.

Large language models (LLMs), trained on vast amounts of textual data, have demonstrated exceptional capabilities in understanding and executing user instructions. Researchers have been leveraging LLMs in a variety of innovative contexts, including image reasoning and editing [5], [6], [7], robotic manipulation [8], [9], mesh and scene generation [10], [11], [12], [13]. These works utilized manually crafted prompts and well-designed multi-agent collaboration architectures, achieving strong performance, and current automatic prompt generation methods [14] still cannot be applied to 3D scene scenarios. Despite their potential, three main challenges hinder the application of LLMs to modeling tasks. First, the unstructured textual outputs generated by LLMs cannot be directly parsed or processed by machines. To address this, we propose a novel JSON-based representation for modeling actions, which allows seamless parsing, interpretation, and execution by downstream systems. Second, traditional interactive modeling pipelines heavily rely on numerous object placement operations, which are difficult to express precisely through natural language. However, textual interfaces excel at specifying high-level modeling logic, such as defining procedural rules and adjusting their parameters. This observation naturally aligns LLM-driven modeling with a procedural modeling paradigm, enabling more rapid and flexible population of elements. Third, relying solely on text-based interactions between users and LLMs introduces inefficiencies in the modeling workflow, particularly when users need to perform frequent, fine-grained manipulations. To overcome this, we propose designing enhanced interaction mechanisms and a modeling memory to refine the modeling experience and improve modeling efficiency.

Donggang Jia and Ivan Viola are with the Computer, Electrical and Mathematical Sciences and Engineering (CEMSE) Division, King Abdulah University of Science and Technology (KAUST), Saudi Arabia. E-mail: {donggang.jia | ivan.viola}@kaust.edu.sa.

Yunhai Wang is with the School of Information, Renmin University of China, China. E-mail: wang.yh@ruc.edu.cn.

Ivan Viola and Yunhai Wang are co-corresponding authors.

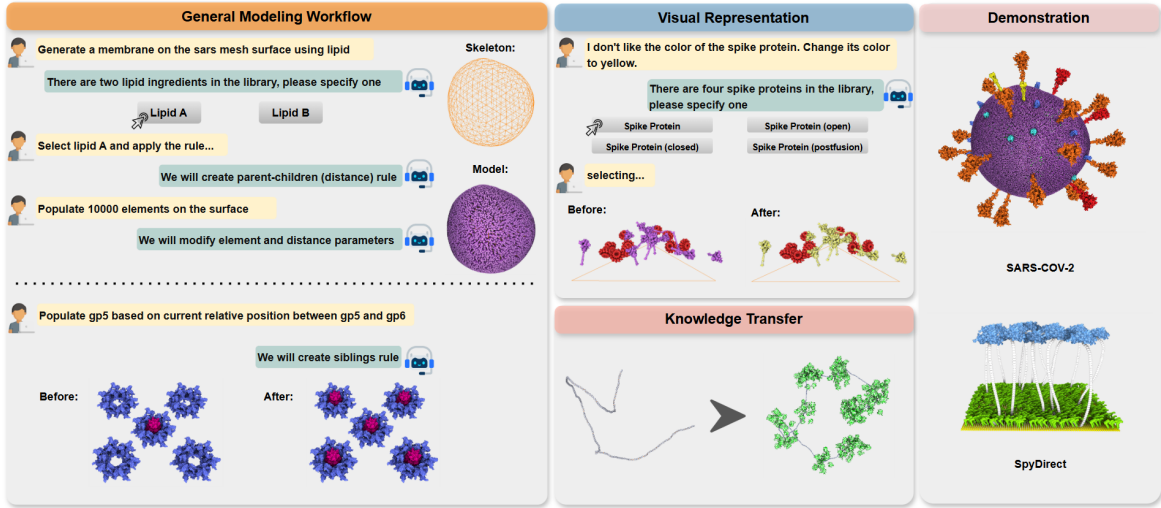


Fig. 1: Chat Modeling is an interactive agent framework that takes users’ multimodal input and completes modeling operations. It can complete the general modeling workflow (left), modify visual representations, or transfer previous modeling templates to a new model (middle). We demonstrate our prototype through two complex biological structures modeled by different modeling modes (right).

In collaboration with structural biologists, we conduct a formative study to gain insights into their typical modeling workflows and their specific needs for a natural language interface tailored to 3D modeling. Based on these insights, we propose Chat Modeling, a procedural modeling agent framework. This framework uses a set of specialized agents to interpret user inputs and carry out specific 3D biological structure modeling tasks, producing both a structured JSON recipe that controls the 3D scene and the corresponding updated 3D models. Each specialized agent focuses on one task and collaborates with the others. To better leverage user data, the framework incorporates a modeling memory designed to improve output accuracy, minimize redundant operations, and make effective use of previous user actions. To further enhance user interaction, we introduce several approaches within the modeling process.

We evaluate our framework through two approaches. First, we construct a benchmark dataset comprising 200 prompts to evaluate the framework’s ability to accurately interpret user commands into modeling operations. The results show that Chat Modeling outperforms general-purpose large language models in this task. Second, we develop a proof-of-concept prototype and conduct both exploratory user study and expert interview to collect qualitative feedback. These evaluations demonstrate the potential of Chat Modeling in supporting biological modeling tasks, underscoring its effectiveness in streamlining the modeling process and allowing users to focus on the creative and analytical aspects of their work.

Our contributions can be summarized as follows:

- We present a conversational agent framework for mesoscale biological modeling that uses scientific literature to extract modeling parameters and supports interactive, step-by-step construction of biological structures.
- We design a new modeling recipe for procedural modeling, together with a corresponding interpreter that transforms users’ high-level and abstract instructions into atomic modeling actions within the modeling software.
- We incorporate task-specific support mechanisms for biological modeling, including modeling memory, plan-stage interaction, and intent-conditioned widgets, to reduce user

burden beyond text-only natural language interaction.

2 RELATED WORK

2.1 Natural Language Interfaces for Visualization

Natural language interfaces have been widely studied as a way to lower the operational and learning barriers of visualization and modeling tools. Existing work spans a range of tasks, from querying structured data and editing visualizations to generating executable programs for visual reasoning, 3D scene authoring, and scientific visualization. A first line of work focuses on natural language interfaces for structured data visualization and visual analytics. Narechania *et al.* [15] proposed NL4DV, which takes tabular data and natural language queries as input and returns an analytic specification in JSON format. Mitra *et al.* [16] extended NL4DV to support multi-round conversational interaction through a conversation manager and query resolver. Wang *et al.* [17] further studied natural language interfaces for visualization authoring, mapping user editing intents to concrete visualization editing actions through an interpreter that combines data-entry abstraction with a deep learning model. More recent systems use large language models to translate high-level language into executable scripts or modular programs for visualization and visual reasoning. Maddigan *et al.* [18] proposed Chat2VIS, which converts user queries and tabular data into Python scripts for chart generation. Gupta *et al.* [6] proposed VISPROG, where an LLM generates Python-like modular programs to invoke existing computer vision models for complex visual tasks. A more closely related direction connects natural language to 3D scene generation, mixed reality authoring, and scientific visualization. Jia *et al.* [19] proposed VOICE for scientific communication, which processes speech input to answer biological questions and generate visual explanations of biological structures. Sun *et al.* [11] introduced 3D-GPT, which uses LLMs to generate Python-like code for invoking functions from Infinigen [20] for 3D scene generation. De La Torre *et al.* [21] proposed LLMR for LLM-assisted scene creation and modification in interactive mixed reality experiences. While these works move closer to spatial and 3D authoring, they primarily focus on scene composition, mixed

reality content, or scientific communication. In contrast, biological modeling requires composing known biological components under literature-grounded, geometric, and quantitative constraints. Our work therefore focuses on a conversational agent framework that connects natural language, literature-derived evidence, and executable modeling operations for biological structures.

2.2 Procedural Modeling

Procedural modeling is an automation technique that generates complex 3D geometries and scenes through algorithms and rules rather than direct manipulation. Since its inception, procedural modeling has been extensively applied in fields such as computer graphics, game development, architectural design, and scientific visualization. The core advantage of this method lies in its efficiency in creating detailed and complex models, particularly suited for scenarios that require repetitive elements and structures, such as plants [22], [23], [24], cloudscape [25], roads [26], street networks [27] and buildings [28], [29].

Raistrick *et al.* [20] introduced Infinigen, a procedural system capable of generating photo-realistic 3D scenes of the natural world. This system can produce infinite shapes, materials, and scene compositions. Each asset, ranging from shapes to textures, is entirely procedural, created from scratch using randomized mathematical rules. These rules facilitate infinite variation and composition, generating highly diverse and unique scenes. Nguyen *et al.* [30] introduced a scientifically accurate procedural modeling method for creating 3D models of cellular mesoscale. Their method is implemented in MesoCraft, a desktop or web-based tool¹ for rapid modeling and visualization of sub-micron biological systems. Mesoscale biological modeling is particularly suited to procedural modeling due to the inherent repetitive elements and specific geometric relationships characterizing these structures.

The steep learning curve of mastering complex geometric definitions in 3D procedural modeling, combined with intricate GUI operations, significantly hinders the user's modeling workflow. Therefore, this paper focuses on facilitating the modeling process by extracting explicit geometric definitions from users' natural language inputs and implementing this workflow in the desktop version of MesoCraft.

3 FORMATIVE STUDY

Our primary objective is to develop a natural language interface to aid users in modeling intricate biological structures. This tool is designed to receive users' inputs, assist them in formulating modeling tasks, and execute the corresponding modeling operations. It is specifically tailored for structural biologists who are new to specific modeling software and often encounter challenges while modeling complex biological structures. Modeling software typically incorporates complex graphical user interfaces and necessitates prior knowledge of spatial and geometric concepts. This combination leads to a steep and time-consuming learning curve for mastering modeling operations, underscoring the necessity for a tool that assists users by simplifying software interactions. A tool designed to streamline the modeling process would significantly benefit users. It would enable them to focus more on their work's creative and analytical aspects rather than on overcoming technical barriers.

To accurately identify the essential features our tool must incorporate, we have consulted with three experts (E1, E2, and E3), all of whom have prior experience in modeling complex biological structures. All experts possess backgrounds in biology, while E3 has additional computer science expertise. Notably, all three experts have experience modeling biological structures with MesoCraft, as demonstrated in previous works [30], [31], [32]. We conducted individual interviews with each expert to understand the specific needs our tool should meet comprehensively.

Each interview was conducted over a span of approximately 90 minutes. Initially, the experts were asked to describe their previous modeling processes, enabling us to grasp their standard modeling workflows and the primary challenges they face. Following this, we delved into understanding their internal logic when determining rule types and asked the experts to construct example models in MesoCraft. The final segment of the interview focused on envisioning the ideal functionalities of a natural language interface designed to facilitate their modeling process. The experts were encouraged to specify the features they believe such a tool should offer and provide examples of how they would articulate these commands in natural language. We report below the design requirements from the interviews.

3.1 Design Requirements

DR1—Consistent with typical modeling pipelines: *The interface should align with experts' typical modeling workflows.* Expert interviews indicate a consistent two-step process. First, users *create rules* (e.g., "Create a lipid bilayer membrane on the virus skeleton surface"). Given that biological ingredients often have similar names and users may prefer not to specify the full names, there is an expectation to intelligently narrow down the options and present a selection of choices for the user to choose from. Second, users *adjust rule parameters* in several iterations, such as 'I would like to populate 1000 lipids on the skeleton surface,' explicitly setting the count while implicitly setting the lipid-skeleton gap to 0. Separating rule creation from tuning reflects real practice, where frequent parameter tweaks are common.

DR2—Iterative rule refinement: *Users can reapply any rule at any time to tweak its parameters and freely reorder its execution within the rule list.* Biological models usually combine many rules, and their execution order is critical: ingredients occupying key positions must be placed first to avoid collisions. Hence, the interface must let users reorder or recall rules at will, enabling precise, incremental refinement of the structure.

DR3—Long-term refinement & reusability: *The interface should be adaptive to user habits and reuse similar operations.* After providing the pairs of natural language inputs and modeling rules, experts indicated that this collected data might not encompass the full spectrum of modeling scenarios or the variety of language expressions used. Structural biologists may lack a comprehensive understanding of geometric positional relationships or possess their own unique terminology for describing these rules. They suggested an enhanced approach: collecting specific user cases and directly incorporating an approval or correction mechanism into the tool. Also, experts further emphasized that, unlike general 3D modeling, which usually creates novel objects tailored to diverse user requirements, biomolecular modeling is driven by experimentally determined or literature-sourced structural knowledge and focuses on faithfully reconstructing known biological assemblies. Because many biomolecules share recur-

1. <https://mesocraft.kaust.edu.sa>

ring structural motifs, the ability to efficiently reuse previously constructed models is especially critical.

DR4–Multi-stage modeling workflow: *The interface should support a clearly structured, multi-stage modeling workflow that makes high-level biological modeling tasks executable and controllable.* Rather than executing a single user-issued command, the system should decompose user-derived intentions into an explicit sequence of modeling operations. While the idea of deriving planning from high-level descriptions has been explored in prior natural-language-driven systems, this requirement centers on adapting such capabilities to the particular demands of biological modeling, such as extracting modeling evidence from publications, generating a high-level modeling plan, exposing intermediate plans for user inspection, and supporting validation and refinement of the resulting model. In our context, the interface should surface these otherwise implicit steps as revisable stages in a structured workflow. This organization helps reduce the need for users to specify low-level geometric details and makes it easier to manage and refine complex procedural modeling pipelines.

DR5–Multimodal interaction workflow: *The interface must go beyond text-only interaction, harnessing the complementary strengths of mouse and text input.* Mouse actions excel at coarse spatial manipulations—such as roughly rotating or translating a structure—whereas text commands provide precise control (e.g., “rotate 30° to the right”) and allow users to issue high-level instructions that bundle several elementary operations. By dynamically selecting the modality that minimizes user effort for each task, the system streamlines the modeling workflow and aligns with users’ natural working habits.

DR6–Multimodal input: *The interface should support multimodal input tailored to biological modeling workflows.* In conventional 3D scene modeling tasks, users typically construct scenes based on their imagination, focusing on the shapes of specific objects and performing multiple rounds of adjustments. Sketches and text are often combined to describe the desired object shapes. In contrast, 3D biological modeling aims to reconstruct known biological structures using modeling tools. In this context, the shapes and approximate locations of proteins (e.g., within or on the membrane) are already determined. Users are more concerned with information such as the quantity of proteins and their spatial distributions, which are often obtained from experiments. Therefore, experts suggest that, beyond traditional text input, the agent is better to support extracting critical data, such as protein counts, from publications like scientific papers, enabling richer and more versatile multimodal input.

To make the relationship between the design requirements and our contributions explicit, we summarize how each requirement informs the proposed system design and corresponding contributions. DR1 and DR2 motivate the design of our procedural modeling recipe and interpreter, which separate rule creation from parameter tuning and allow rules to be reapplied and reordered during modeling. DR3 directly motivates the customized modeling memory, which captures user preferences, collects feedback, and enables reuse of prior modeling operations in literature-grounded biological modeling tasks. DR4 motivates the overall conversational agent framework and the explicit decomposition of high-level biological intentions into a structured, inspectable, and revisable plan. DR5 and DR6 motivate our interaction designs during plan execution, including intent-conditioned widgets and the integration of scientific literature as an input modality.

ALGORITHM 1: Chat Modeling

Input: u : user input; Ω : current scene raw; m : modeling memory;

Require: $S(s \mid \Omega)$: Scene Summarizer

$P(u_1, \dots, u_N \mid u, s, m)$: Modeling Planner

$G(j \mid u, e, m)$: Recipe Generation Agent

$C(e \mid j)$: Recipe Correction

$I(\mathcal{R}, \mathcal{P}, \mathcal{A} \mid j, m)$: Recipe Interpreter

```

1  $s \sim S(\cdot \mid \Omega)$ 
2  $(u_1, \dots, u_N) \sim P(\cdot \mid u, s, m)$  //generate plan from multi-turn
   dialogue
3 for  $i \leftarrow 1$  to  $N$  do
4    $j \sim G(\cdot \mid u_i, m)$  //generate recipe based on the step
5    $e \sim C(\cdot \mid j)$ 
6   while  $e$  do
7      $j \sim G(\cdot \mid u_i, e, m)$  //correct json by the error info
8      $e \sim C(\cdot \mid j)$  //check the error again
9   end
10   $\mathcal{R}_i, \mathcal{P}_i, \mathcal{A}_i \sim I(\cdot \mid j, m)$  //Interpret the recipe to operations
11   $\mathcal{M} \leftarrow \text{RunInMesoCraft}(\mathcal{R}_i, \mathcal{P}_i, \mathcal{A}_i)$ 
12 end

13 Notation:  $u_i$  denotes the  $i$ -th step in the modeling plan;
14  $j$  is a structured recipe representation;
15  $e$  indicates validation errors;
16  $\mathcal{R}, \mathcal{P}, \mathcal{A}$  denote modeling rules, parameters, and actions,
   respectively.
```

4 METHOD OVERVIEW

Recent advances in LLMs have greatly improved natural language understanding and instruction following capabilities. Leveraging these capabilities, we aim to propose an agent framework that closes the gap between free-form textual commands and the mouse-driven workflows of existing modeling software. Informed by DR4 (multi-stage modeling workflow) from our formative study, the framework not only assists users in devising a step-by-step modeling plan but also translates each planned step, or raw user instructions, directly into concrete modeling operations.

An effective strategy to bridge this gap involves adopting the procedural modeling paradigm and creating an intermediary component layer. This layer serve as a conduit between users and the modeling software. Essentially, it would translate human-readable instructions into a software-processable recipe, acting as a translator between the user’s textual descriptions and the software’s command syntax. Here, a recipe denotes a structured model specification—listing the biological components, their quantities, and relevant constraints/parameters—that the modeling system can directly execute.

The overview of our framework is illustrated in Figure 2 and formatted in algorithm 1. Our workflow starts with a user’s input u . To fulfill DR6 (multimodal support) that biological modeling always aims to reconstruct known biological structures, we incorporate two types of input: textual descriptions and scientific publications such as research papers. User input together with the scene information s , is sent to the modeling builder, the bridging layer between the user and the MesoCraft. We define s as the current scene state, which consists of a text-based description of the objects, and structures already present in the scene. The modeling builder performs two complementary tasks: (1) through multi-turn dialogue, it distills the user’s description into an ordered, high-level modeling plan; and (2) it converts

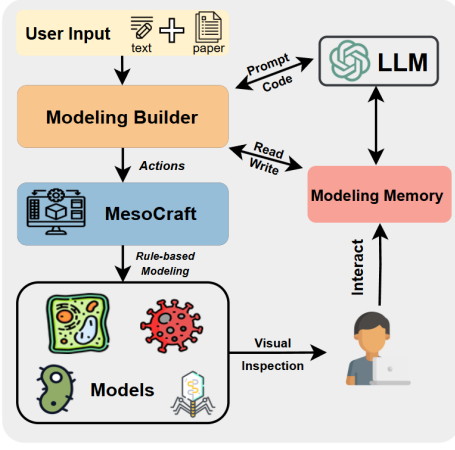


Fig. 2: The framework starts with user input, processed by the Modeling Builder to generate modeling plans. MesoCraft then models biological structures from these plans. Users visually inspect results and interact with the modeling memory to improve the whole framework.

each plan step u_i into machine-readable JSON recipe j that is subsequently translated into native MesoCraft operations, such as rules $\mathcal{R}_i$, parameters $\mathcal{P}_i$ and modeling actions $\mathcal{A}_i$. MesoCraft then executes these operations, generating the requested biological structures $\mathcal{M}$ for the user’s inspection and refinement. To enable continual improvement, the framework also features a dedicated modeling memory m that aggregates data from both the user’s visual feedback and the dialogue history to enhance planning and execution quality across future modeling sessions.

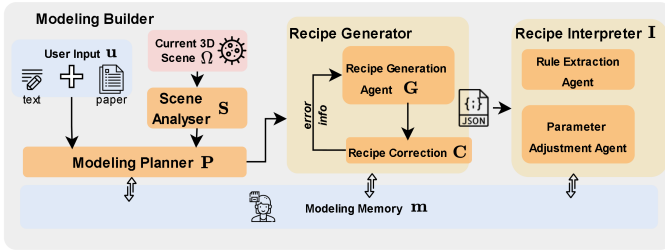


Fig. 3: The modeling builder consists of the modeling planner, recipe generator, and recipe interpreter. The recipe generator creates validated recipe based on the plans from the modeling planner, while the recipe interpreter converts this recipe into procedural modeling actions.

The modeling builder, illustrated in Figure 3, consists of three principal components, *i.e.*, the modeling planner P , the recipe generator and the recipe interpreter I . The modeling planner takes user input u and scene description s as input and generates a modeling plan $U = \{u_1, \dots, u_N\}$, where each u_i represents an individual modeling step. The recipe generator transforms steps into validated and executable recipe snippets j . Following this, the recipe interpreter I parses and interprets these validated scripts. It is crucial in translating abstract recipe into concrete operations within MesoCraft, effectively bridging the gap between user intent and software execution.

The recipe generator component includes two key operations: recipe generation agent G and recipe correction operation C . Utilizing LLMs, it deciphers user commands to produce accurate

recipe snippets encapsulated in a novel JSON format, tailored for complex biological modeling workflows. After recipe generation, the recipe correction process ensures the accuracy and effectiveness of these snippets by feeding any detected errors back into the LLMs for modification. The recipe interpreter involves three main components: a rule extraction agent, which identifies rule types $\mathcal{R}_i$ based on the geometrical and spatial relationships in the step; parameter adjustment agent, which allows for fine-tuning these rule parameters $\mathcal{P}_i$; and functionalities for visual representation modifications and other essential actions $\mathcal{A}_i$ within the general modeling workflow. Once interpreted by the recipe interpreter I , the actions are executed within MesoCraft to create complex biological models, which are then presented back to users.

As mandated by DR3 (long-term refinement and reusability), structural biologists articulate geometric and spatial relationships in highly diverse ways, which can reduce the reliability of the framework’s natural-language-driven modeling operations. To accommodate this variability, we embed a modeling memory into our workflow. The system persistently logs users’ interactions within the modeling builder and records their visual-inspection feedback on generated results. These traces are subsequently leveraged to iteratively refine future outputs and to recall previously validated operations, thereby lightening users’ modeling workload and enhancing overall accuracy.

To address the issue of text-only interactions being suboptimal for certain scenarios, as outlined by DR5 (multimodal interaction workflow), we incorporate multiple interaction strategies during the plan execution stage. This hybrid approach enhances the modeling process, reducing user effort by enabling more intuitive and efficient interactions.

5 MODELING BUILDER

This section details the implementation of each component within the modeling builder, which generates the modeling plan and translates natural language steps into executable actions of the modeling software.

5.1 Memory-enhanced Few-shot Prompting

As outlined in section 4, our approach leverages LLM-based agents to handle various tasks in the modeling builder. While LLMs excel at grasping general semantics, their application in task-specific scenarios requires good prompts. Prompting is a technique that involves the careful crafting of inputs to the model to enhance task performance. Importantly, this process does not involve modifying the model’s weights, ensuring that the LLM can be effectively utilized for specific tasks through strategic input manipulation alone. Our methodology adopts a memory-enhanced few-shot prompting setting, demonstrated in Figure 4. The prompt contains a concise task description and a few task-specific examples. Previous work [33] verified that a few examples could significantly improve LLM performance. We collected these examples from experts in the formative study.

To accommodate the diverse ways in which users articulate modeling intent, and to enable our framework to improve its accuracy over time, we append a user-specific, dynamic memory block to the prompts. This block either supplements the task-specific examples discussed above or encodes habitual modeling preferences unique to the current user. As the modeling memory is updated and queried, the contents of this block are updated periodically, allowing each prompt to be continuously tailored

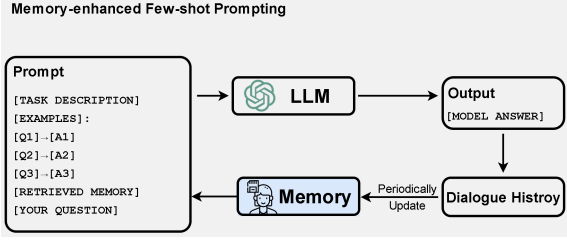


Fig. 4: Memory-enhanced few-shot prompting setting. The prompt includes a task description, initial examples, and retrieved memory. Memory is periodically updated from the dialogue history.

to the user’s needs. The design of the memory component is described in detail in [section 6](#), and complete prompt templates of different agents are provided in the supplementary material.

5.2 Modeling Planning

The modeling builder initiates with a planning stage that integrates the user’s multi-turn dialogue with the current on-screen context to generate a detailed modeling plan. This plan serves as the blueprint for downstream agents to construct the target biological structure. The planning stage involves two cooperative agents: the scene summarizer agent, which converts raw screen data into a textual scene description, and the modeling planner agent, which fuses the user’s dialogue with the scene description to produce the finalized modeling plan.

5.2.1 Scene Summarizer

Our modeling planning stage operates on the current scene data, which stores heterogeneous information such as the name, category, and spatial pose of every instance. Because these attributes are scattered throughout complex scene data, we introduce a scene summarizer agent S that converts the raw scene data Ω into a concise textual summary of scene information s for the planner. Concretely, the raw scene data Ω is extracted from the MesoCraft scene as a JSON object that lists essential information such as each instance’s name, type, position, rotation, together with the global skeleton and per-instance hierarchy metadata. When a biological structure contains numerous copies of the same protein, the raw scene data Ω omits per-instance pose details (e.g., exact positions and orientations) to keep the summary compact. One example of the input and output of the scene summarizer agent is shown in [Figure 5](#). The detailed agent prompt is provided in supplementary section 2, which includes the task description and several illustrative examples.

Input: {objects:[{name: Cube,type:skeleton,edge_length:100},{name:Cube-Fill,type:model,children:[{name:spike protein,type:protein,position:(10,20,30),rotation:(0,0,0,1),children:[]}, {name:spike protein,type:protein,position:(-15,12,13),rotation:(0,0,0,1),children:[]}]}]}

Output: The scene contains a Cube skeleton and a model named Cube-Fill, the model contains two spike proteins.

Fig. 5: An example illustrating the input and output of the Scene Summarizer, which converts raw scene data into semantic scene information.

5.2.2 Modeling Planner

In traditional mouse-driven GUI-based workflows, users need to first conceptualize the model’s global structure, shape, and the spatial relations among its constituent instances. Powered by the broad biological knowledge embedded in LLMs, we introduce a modeling planner agent P to fulfill DR4 (multi-stage modeling workflow). Through a multi-turn dialogue, the planner, grounded in the current scene context s , returns an initial modeling proposal. The user may then iteratively revise this proposal until a finalized plan is obtained.

$$P(u_1, \dots, u_N | u, s, m) = \prod_{i=1}^N P(u_i | u, s, m, u_1, \dots, u_{i-1}), \quad (1)$$

where u denotes the user’s query, s the textual scene summary, and m the memory. The variable u_i represents the i^{th} step proposed from the plan, and the planner’s output is the sequence $U = \{u_1, \dots, u_N\}$. As specified by DR6 (multimodal input), 3D biological modeling focuses on reconstructing known biological structures, with particular emphasis on the quantity of proteins and their spatial distributions, which are of primary concern to users. Consequently, the user query u consists of two components: textual descriptions provided by the user, and scientific publications such as research papers. The scientific publications, which are optional, are primarily used to supply accurate parameters. In their absence, the parameters are inferred either from the LLM’s internal knowledge or directly from user input. To personalize planning, the memory m encodes user-specific context such as modeling preferences, enabling the planner to adapt its suggestions to the current user. When scientific publications are unavailable or insufficient, parameters may be inferred from the LLM’s internal knowledge, which introduces the risk of hallucination. To mitigate this issue, all parameters inferred from literature are explicitly marked in the dialogue interface, making their provenance transparent to the user. Importantly, these inferred parameters are treated as tentative initial conditions rather than fixed constraints: they only initialize the planning process and can be inspected, modified, or overridden by the user through subsequent conversations. This design enables efficient human verification and correction, ensuring that potential hallucinations do not propagate into the final modeling results.

The detailed modeling planner agent prompt is provided in supplementary section 1, which includes the task description, detailed guidelines, and the basic logic of the MesoCraft modeling process, current scene information, user modeling preference, and a few modeling plan examples. The current scene information is provided by the scene summarizer agent, and the user modeling preference is from the modeling memory.

5.3 Recipe Generator

The recipe generator comprises two key operations, *i.e.*, recipe generation agent and recipe correction operation. Through iterative adoption of these operations, the recipe generator forwards validated recipes to the recipe interpreter for further processing.

5.3.1 Recipe Generation

The recipe generation operations extract user intents into machine-readable recipes, aiming to precisely mirror the user’s modeling workflow actions and accommodate the requirements for visual representation modifications. The chosen recipe format must be

easily parsed and executable for subsequent interpretation. Based on these criteria, we propose a new JSON format covering user intents collected from the formative study. JSON is a standardized yet flexible format, ideally suited to handle the complexities of modeling tasks. The specifics of the recipe format are illustrated in [Recipe Format 1](#).

Our format incorporates multiple intents to support the general modeling workflow within MesoCraft comprehensively, as requested by DR1 (consistent with typical modeling pipelines). Each modeling action is predicated upon creating and applying specific rules. These rules comprise one or several ingredients and an optional skeleton structure to guide the geometrical arrangement of populated instances. We introduce two pivotal intents, *i.e.*, *selectIngredient* and *selectSkeleton*, to support modeling basic operations. Because a single instruction may reference multiple items, both accept arrays. We also add *createRule* and *editRule* for creating new rules and modifying parameters. Given the difficulty of inferring rule types and numerous tunables, the recipe interpreter includes a rule extraction agent and a parameter adjustment agent. The recipe retains rule and parameter descriptions for downstream processing. Finally, *saveModel* and *loadModel* are used for model save and load. Natural language conveys precision that mouse clicks cannot, for example, positions by amino acid index in biology. To leverage this, we add *updatePivot* and *updatePosition*: users can set an instance’s pivot by amino acid index and update its position by specifying spatial relations between amino acids across instances. For visual representation edits (informed by our formative study), we include *highlightIngredient*, *modifyColor*, and *changeMode*. *highlightIngredient* emphasizes selected ingredients; *modifyColor* alters their colors; both accept arrays to batch multiple changes per instruction. *changeMode* switches MesoCraft rendering among protein, chain, and atomistic-level views, each offering a distinct perspective for analysis or presentation.

```
{
  "selectIngredient": [
    { "ingredient": (Str) ingredient_name },
  ],
  "selectSkeleton": [
    { "skeleton": (Str) skeleton_name },
  ],
  "createRule": (Str) descriptions_to_the_rule,
  "editRule": (Str) descriptions_to_edit_the_rule_parameters,
  "saveModel": (Bool) if_save_model,
  "loadModel": (Bool) if_load_model,
  "updatePivot": {
    "chainId": (Int) chain_index,
    "residueId": (Int) residue/amino_acid_index,
  },
  "updatePosition": [
    { "mainIngredient": (Str) ingredient_name,
      "chainId": (Int) chain_index,
      "residueId": (Int) residue/amino_acid_index,
      { "subIngredient": (Str) ingredient_name,
        "chainId": (Int) chain_index,
        "residueId": (Int) residue/amino_acid_index },
    ],
  ],
  "highlightIngredient": [
    { "ingredient": (Str) ingredient_name },
  ],
  "modifyColor": [
    { "ingredient": (Str) ingredient_name,
      "color": (vector3D) ingredient_color },
  ],
  "changeMode": (Str) rendering_mode_name_to_change,
  "labeling": (Bool) if_labeling_enabled }

```

Recipe Format 1: Syntax format of the recipe generation agent.

The detailed recipe generation agent prompt is provided in supplementary section 3, which includes the entire task description, detailed JSON format, reasoning logic about the recipe generation, and a few recipe generation examples together with the user feedback from the memory system.

5.3.2 Recipe Correction

We incorporate a recipe correction stage into the generator to handle small but consequential errors in LLM outputs. Although

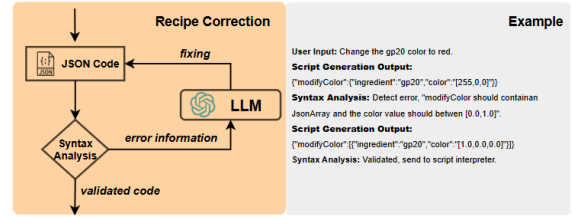


Fig. 6: The recipe correction operation involves an iterative process for syntax analysis and error fixing.

recent LLMs perform well at recipe generation [34], domain-specific constraints, such as our custom schema for complex biological models, still lead to occasional format discrepancies. As shown in [Figure 6](#), we implement a syntax analysis module in MesoCraft, containing both heuristic rules and a JSON schema linter to check the recipe code. This module functions as a quality-assurance gate that enforces strict conformance before any recipe reaches the interpreter: it reviews each recipe for parseability (syntactic validity), validates array-object structure for intents such as *selectIngredient* and *modifyColor*, and verifies that all values match the predefined data types in our schema. When violations are detected, we attach concise, manually defined error descriptors and return the recipe to the generator for up to $n=2$ revision rounds; each round addresses only the reported issues and then re-runs validation. Once all checks pass, the corrected recipe proceeds to the interpreter. The example on the right of [Figure 6](#) shows two syntax errors that are flagged, corrected in one pass, and subsequently validated.

5.4 Recipe Interpreter

Upon receiving the validated recipe, the recipe interpreter parses it and executes it into specific actions within MesoCraft. Actions like color modification and ingredient selection are directly executed using the key-value pairs from the scripts. Given the complexity of identifying rule types for rule creation and the necessary parameters for rule editing, we design separate agents, *i.e.*, the rule extraction agent and the parameter adjustment agent.

5.4.1 Rule Extraction Agent

Our framework operates on the rule system of MesoCraft, which represents procedural modeling operations through six classes of geometric rules: *parent-child (distance)*, *parent-child (relative)*, *siblings*, *siblings-parent*, *fill*, and *connection* rules. The rule extraction agent maps high-level user instructions or recipe descriptions to the appropriate rule type and its parameters, so that the modeling intent can be executed by MesoCraft. For example, a parent-child distance rule can place new elements at a specified distance from a skeleton surface, while a siblings rule can infer transformations between existing elements and reuse them to generate additional instances. Other rule types similarly encode geometric relations between elements, skeletons, or trajectories, enabling MesoCraft to populate and organize biological components in a controllable procedural manner. Representative examples are shown in [Figure 7](#).

In MesoCraft, the first step in creating a rule is to determine its class. This choice depends on multiple factors, including the number of ingredients involved, the dimensionality of the skeleton (2D or 3D), and the spatial relationships between ingredients

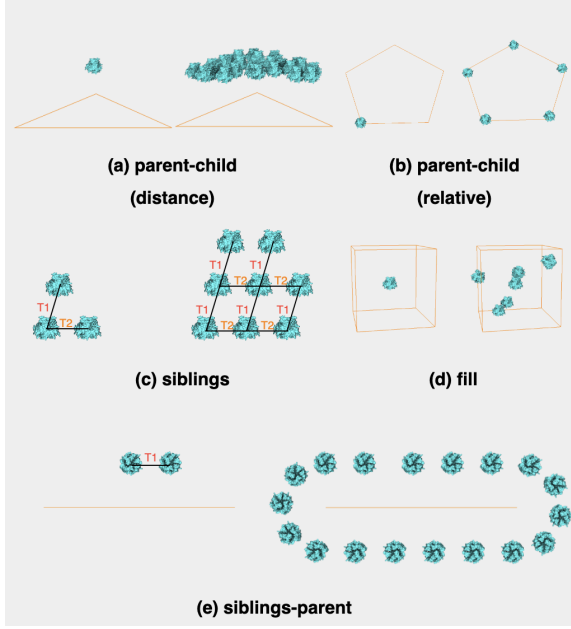


Fig. 7: Examples of rule types: for each rule, the left image shows rule creation and the right image shows the outcome after rule application.

and the skeleton. Because this decision is non-trivial, we treat rule-type selection as a separate subtask and introduce a rule extraction agent within the recipe interpreter. More specifically, the agent’s prompt explicitly encodes the descriptions of each rule type together with the corresponding decision criteria, and is further augmented with system-provided examples and relevant rule-extraction-related user feedback from user memory. This prompt design guides the agent to determine the most appropriate rule class. Given a textual rule description, this agent predicts the appropriate rule class, whose parameters are then refined by the parameter adjustment agent. The detailed rule extraction agent prompt is listed in supplementary section 4.

5.4.2 Parameter Adjustment Agent

An essential operation of the recipe interpreter involves parameter adjustment. As highlighted in DR2 (iterative rule refinement), users often use multiple iterations to fine-tune rule parameters, ensuring the modeling output meets their expectations. The design of the parameter adjustment agent follows the recipe generation paradigm, introducing a novel format to encapsulate the various rule parameters, as depicted in [Recipe Format 2](#). This process begins with processing user instructions to generate a corresponding validated recipe via the recipe generator. Subsequently, the recipe interpreter parses and interprets this validated recipe, facilitating precise adjustments to the model.

```
{
  "elements": (Int) number_of_elements,
  "distance": (Double) distance_to_the_skeleton,
  "collisionDetection": (Bool) if_collision_detection_enabled,
  "space": (Int) occupied_space,
  "alignDirection": (Str) elements_aligned_direction,
  "length": (Double) generated_curve_length,
  "curve": (Str) generated_curve_type,
  "tweaking": (Str) tweaking_directions,
  "std": (Double) standard_deviation_for_the_distance
}
```

Recipe Format 2: Syntax format of parameter adjustment operation.

We encode a compact set of parameters in the recipe format: target rule elements, a collision-detection flag, mean/SD of instance–skeleton distance, alignment (along or opposite the skeleton normal), generated-curve length, and tweak angle. We also include high-level intent into the parameters. For example, a *space* parameter specifies the fraction of the 3D skeleton to occupy; given the skeleton and ingredient volumes (approximating the latter with a bounding sphere), we compute the corresponding element counts and sizes. The detailed parameter adjustment agent prompt is listed in supplementary section 5, which contains a task description, the detailed format of the parameter recipe, and examples collected both from the formative study and user feedback from the modeling memory.

6 MODELING MEMORY DESIGN

The modeling builder supports multi-turn, dialogue-driven biological modeling. To support DR3 (long-term refinement & reusability), we incorporate a lightweight modeling memory that helps the system adapt to user preferences, preserve corrective feedback, and reuse prior operations across sessions. Rather than introducing a new general-purpose memory architecture, our design adapts common memory mechanisms in conversational AI systems [35], [36], [37] to the procedural workflow of biological modeling.

As illustrated in [Figure 8](#), the memory widget organizes four types of information. Modeling preferences store user-specific conventions, such as preferred parameter settings or naming styles. They are extracted from dialogue logs after multiple planning rounds and can be viewed, edited, or deleted by the user. User feedback records explicit corrections reported during plan execution, such as incorrect rule interpretation or parameter adjustment. The system converts such corrections into instruction–action pairs and routes them to the corresponding agent, including the recipe generation, rule extraction, or parameter adjustment agent. Previously executed plans store complete plan descriptions and executable plans as reusable templates for similar biological structures. Chat history maintains planner-relevant conversational context indexed by session IDs, enabling users to resume prior modeling sessions.

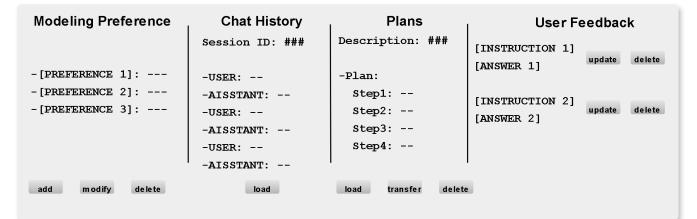


Fig. 8: Demonstration of the memory management widget.

These memory types are consumed differently during modeling. Modeling preferences are appended to the planner prompt to condition planning and parameter selection. User feedback is injected into the corresponding agent prompt as few-shot instruction–action examples. Previously executed plans can be selected as templates to bootstrap new modeling processes, while chat history provides context for continuing prior sessions.

For example, if the recipe generation agent produces a valid JSON recipe but the rule type is misinterpreted by the rule extraction agent, the user can report the error and provide the correct interpretation. The correction is stored as feedback memory for

the rule extraction agent and later reused as a few-shot example when similar rule descriptions appear.

7 USER INTERACTION ENHANCEMENTS

In the preceding sections, we introduce our agent framework grounded in the design of a modeling builder and a modeling memory. The framework is designed to accept text input and research papers as its primary inputs. However, as articulated by DR5 (Multimodal Interaction Workflow), relying solely on text to control the modeling process is often cumbersome. To better leverage the complementary strengths of natural-language specification and mouse-based direct manipulation, we employ a suite of multimodal interaction–augmentation strategies that more effectively support users in completing modeling tasks.

Intent-conditioned widgets: We design interaction modes by estimating the user effort associated with each operation and selecting the modality that minimizes this cost. For instance, continuous geometric adjustments (e.g., translation, rotation, scaling) are mapped to mouse-based direct manipulation. Discrete visual-representation edits (e.g., color) are offered through concise text commands or lightweight form controls, which reduce pointing and menu navigation; once a visual change is triggered via text, the user can continue refining it in a dialog using a color picker without issuing additional textual commands. In the biological domain, where protein names are often similar, we generate selection buttons in the dialog whenever user disambiguation is required, and execution only proceeds after the user makes a selection. To support the DR2 (iterative rule-refinement paradigm) in biological modeling, where each rule undergoes multiple rounds of fine-tuning to meet user expectations, we dynamically generate control buttons when a rule is applied. Users may then adjust rule parameters either through natural-language specification or via interactive widgets in the dialog (e.g., sliders, numeric inputs), thereby minimizing effort. Overall, intent-conditioned widgets allow users to perform modeling operations either purely through natural-language instructions in the dialog or by quickly reapplying the same operation via intent-conditioned widgets. This not only reduces user effort but also ensures that, after multiple iterations, users can conveniently reapply previously executed operations.

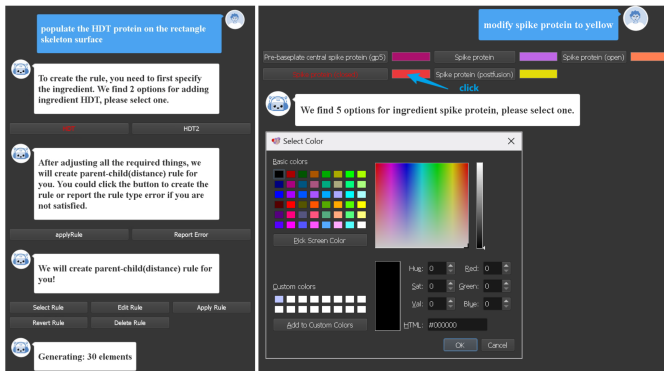


Fig. 9: Examples of intent-conditioned widgets. Left: the system generates selection buttons to disambiguate proteins with similar names, as well as control widgets for iterative rule refinement. Right: a visual attribute (color) is first modified through a text instruction and can then be refined using a dynamically generated color picker.

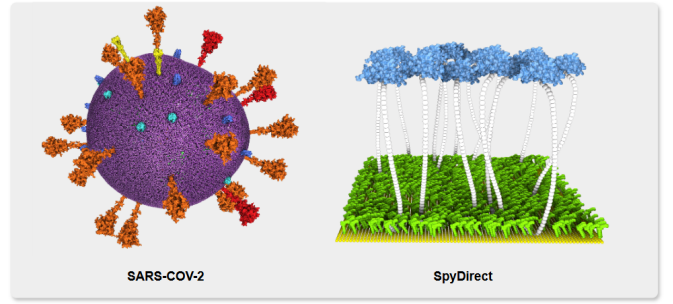


Fig. 10: Demonstration models of our two modeling modes.

Different modeling modes: Our Chat Modeling supports two complementary modes that reflect how users actually work and the varying planning overhead they are willing to incur. First is the planning mode. Through multi-turn dialogue, the system and user co-author a structured plan. Before execution, the plan is reviewable and editable; this mode suits complex or uncertain tasks where users benefit from receiving the overall strategy from the framework. Second is the step-by-step mode. When users already have a clear plan by themselves but are unfamiliar with the GUI or prefer quick, incremental progress, they can issue each operation through language and receive an immediate result. We also design an iterative plan execution approach for biology modeling task. Because biological modeling commonly requires fine-tuning at each stage to meet user expectations (DR2), plans are not executed in a single pass. Instead, execution proceeds in a review–refine–advance loop: after each step, users can still issue new language operations or utilize the widgets in the dialog to fine-tune the model. Until satisfied, users then press *Next* to apply the subsequent step. This interleaving of execution with localized parameter tuning preserves control fidelity while avoiding the overhead of re-planning.

8 DEMONSTRATION

As a proof of concept, we develop a prototype utilizing our framework, Chat Modeling. This section describes two modeling results using our prototype’s different modeling modes. The detailed modeling videos are attached in the supplementary materials.

8.1 SARS-COV-2 Model

Our first case study models a simplified SARS-CoV-2 structure [30], illustrated in Figure 10 left. The user adopts an outside-in modeling strategy: first construct the virus’ outer layer and then proceed to the inner layer. The user begins by planning the outer layer. To determine quantities such as the number of surface proteins, the user uploads a research paper on SARS-CoV-2 [38] and asks how many surface proteins should be modeled for the outer layer. Querying the paper, the *Planner* responds: “A SARS-CoV-2 virion has, on average, 26 ± 15 spike proteins randomly distributed on its surface,” and notes that the paper does not report exact counts for the membrane and envelope proteins. The original wording in the paper is: “An average of 26 ± 15 prefusion S proteins were found to be randomly distributed on each virion.” With the spike count established, the user requests a final plan for the outer layer, approves it, and begins execution. The plan includes modeling spike proteins in three distinct stages, then modeling the membrane and envelope proteins on the surface,

and finally modeling the membrane itself. As the plan is executed, Chat Modeling dynamically generates widgets in the dialogue to support multi-round parameter tuning for each step; the user iteratively adjusts parameters until satisfied and then clicks to advance to the next step.

When preparing to model the membrane, the user realizes that a complete membrane would occlude inner structures. Accordingly, they defer the membrane step, first modeling the inner layer and returning to the membrane afterward. For the inner layer, the user aims to model the RNA strand. Because RNA strand thread through nucleocapsid proteins, the user reuses a previously saved “RNA strand” template from the modeling memory. This template models RNA via control points. Using the transfer-knowledge mode, the user specifies that the positions of nucleocapsid proteins should guide the placement of the RNA strands, producing a new plan: first populate nucleocapsid proteins within the skeleton, then connect positions into RNA strands.

After completing the internal RNA strands, the user relick the previously generated widget for the membrane rule, edits the parameters, and models the full membrane. Notably, the entire workflow is conducted within the dialogue, without hunting through a complex GUI. The intent-conditioned widgets allow rapid refinement at each step; once the inner layer is finished, the user can also revisit earlier widgets to edit prior rules.

8.2 SpyDirect

Through the modeling of SARS-CoV-2, we demonstrate structured planning and the reuse of prior modeling templates. The planning mode is best suited for scenarios in which the user does not yet have a concrete modeling plan. When the user already has a clear plan and does not require agent-driven planning, repeatedly invoking a plan-execute loop can be unnecessarily time-consuming. To illustrate how Chat Modeling supports such cases, we reproduce a simplified, proof-of-concept biological structure following Guo *et al.* [39], illustrated in Figure 10 right, using the system’s step-by-step mode to construct the model directly and incrementally.

The modeling process begins with a bottom gold layer. This layer consists of gold atoms uniformly arranged across a rectangular skeleton. The instruction provided for this task is, “Populate the Au atom uniformly on a rectangle skeleton.” The “fill” rule type is identified through the rule extraction operation. After a single round of adjusting the element parameters, the gold layer is successfully generated. The subsequent step involves generating hexanedithiol (HDT) and SpyCatcher instances above the gold layer. Following the user’s instructions, Chat Modeling creates two parent-child distance rules. Notably, due to the presence of two similar HDTs in the software, it’s necessary for the user to specify the precise HDT variant. This functionality is particularly crucial in biological modeling, where a single protein may have multiple variants across different stages, each with similar names but distinct structural properties. For each parent-child distance rule, the user takes multiple rounds to adjust the rule’s parameters, such as element number, the average and standard deviation of the distance, and the tweaking directions. The final step in the modeling process involves creating the protein linkers between HDT and SpyCatcher instances. A protein linker is a short sequence of amino acids used to connect two proteins or domains within a single protein. In our modeling process, we use small balls to represent the amino acids. This protein linker creation process comprises two main steps: initially creating curves between the

TABLE 1: Numerical results of different methods on our dataset. We report the average accuracy and standard deviation over five runs.

Method	GPT-4o		GPT-4-turbo	
	Avg. Acc.	Std.	Avg. Acc.	Std.
Zero-Shot	0.505	0.008	0.420	0.020
Few-Shot	0.771	0.029	0.738	0.011
Our Method w/o Recipe Correction	0.847	0.019	0.810	0.018
Our Method	0.918	0.011	0.939	0.014

HDT and SpyCatcher instances and then populating these curves with balls. The user then changes the pivot point of SpyCatcher to adjust the curve’s starting point using a specific chain ID and residue ID. Subsequently, curves are formed through the application of a connection rule. The user then proceeds to populate these curves with balls and adjust the number of balls.

9 RESULTS

In this section, we present the quantitative results of our step execution performance on a dataset of task-annotated JSON recipe pairs and discuss the implementation details of Chat Modeling.

9.1 Quantitative Results

To evaluate the effectiveness of our framework, we created a dataset of 200 task-annotated JSON recipe pairs. These tasks were initially gathered from the formative study and were further expanded to encompass a variety of modeling operations, including rule creation, rule parameter adjustment, and visual representation modification. Some tasks involve multiple types of operations; for example, “populate thirty capsid proteins inside the capsid” combines rule creation and editing. Detailed tasks are provided in the supplemental material.

We tested our step execution approach (involving recipe generator and recipe interpreter) on two large language models, GPT-4o [40] and GPT-4-turbo [41], and compared it to three model settings: zero-shot, few-shot, and our method without recipe correction. We selected accuracy as our evaluation metric. Our correctness criteria are as follows: the format should be consistent with the annotations, and the values of the key-value pairs in the JSON recipe must be accurate. Certain aspects, such as whether an integer is returned as int or float, or minor discrepancies in color values (as some colors don’t have standardized RGB values), are acceptable deviations. To ensure a fair quantitative evaluation, we did not involve any modeling memory in any of the settings. The results are summarized in Table 1.

In both zero-shot and few-shot settings, we unified the recipe generator and recipe interpreter within a single prompt, deliberately not employing the recipe correction operation. In the zero-shot setting, no examples are provided, whereas the few-shot setting includes several examples within the prompts. Our findings indicate that the few-shot setting surpasses the zero-shot setting by offering illustrative examples. Furthermore, our method, which excludes recipe correction, outperforms the few-shot setting by completing modeling tasks within several steps. Notably, our approach achieves the highest performance when incorporating recipe correction, thereby enriching the information available for the LLM’s reasoning process. These results underscore the effectiveness of our method in completing modeling tasks.

To further evaluate the effectiveness of modeling memory, we conducted a small-scale memory ablation study using 40 sampled tasks from the evaluation set tested on the recipe generation agent. In the baseline condition, the recipe generation agent was prompted in a zero-shot manner without any examples. In the memory condition, we simulated a user-feedback scenario by injecting only one memory example for each task, consisting of the same task query and its corrected recipe output. This setting evaluates whether the system can reuse previously corrected feedback to improve subsequent recipe generation. Using the same correctness criteria as above, GPT-4o improved from 47.5% accuracy in the zero-shot baseline to 92.5% with memory, while GPT-4-turbo improved from 45.0% to 97.5%. These results provide empirical evidence that the modeling memory can substantially improve recipe generation accuracy after user feedback is incorporated.

9.2 Implementation Details

We implement Chat Modeling as a plug-in for the desktop version of MesoCraft, allowing the framework to directly invoke its procedural modeling and visualization pipeline. The prototype uses a chat-based interface connected to an intermediate visualization-action layer, which translates user instructions into structured recipes encoding executable modeling and visualization operations. These recipes are validated before execution to ensure schema consistency and domain-specific constraints, with validation errors returned as textual feedback for correction. The interface also implements the interaction mechanisms described earlier, including intent-conditioned widgets for parameter tuning and representation adjustment, stepwise execution for inspecting intermediate results, and direct mouse interaction for continuous spatial manipulation such as translation and rotation. This implementation enables language, widgets, and direct manipulation to operate within the same MesoCraft workflow.

10 USER EVALUATION

We conducted a two-stage evaluation consisting of an exploratory user study and expert interviews. The exploratory user study focuses on preliminary usability feedback and perceived workload, while the expert interviews assess domain validity, practical usefulness, and real-world applicability from the perspective of biological modeling experts.

10.1 Exploratory User Study

We conducted a user study to compare the user experiences of traditional GUI-based modeling and language-based Chat Modeling. A total of six volunteers (three women and three men), denoted as P1–P6, participated in the study. Participants were aged between 24 and 32 years ($M = 26.8$, $SD = 3.4$) and were all graduate students in biology-related fields with no prior experience using MesoCraft. Participants were compensated with gift vouchers valued at approximately USD 25. The experiment was conducted by trained staff who had completed data security and bioethics training. The study was approved by the institution’s bioethics committee and was carried out at the institution’s facilities.

10.1.1 Procedure

The user study began with a brief introduction to MesoCraft and the Chat Modeling prototype. Participants were then asked to complete a simplified SpyDirect (right side of Figure 10) modeling

- Q1: It’s quick to understand how to perform the modeling tasks.
 Q2: It’s easy to use the system even without prior experience with MesoCraft.
 Q3: The system allowed users to specify modeling operations at a higher, more intuitive level than traditional GUI-based tools.
 Q4: Users could express modeling intents easily using natural language in the system.
 Q5: The system helped users focus more on the modeling goal rather than tool operation.
 Q6: Users could take control of the modeling process when using the system.
 Q7: It’s easy to correct or refine modeling results in the system.
 Q8: The system would be useful for users who are new to biological modeling tools.
 I1: What features are missing in the current system?
 I2: What features of the current system are most impressive?

TABLE 2: Questionnaire and later interview questions.

task, described as follows: “The model includes a gold layer base with hexanedithiol (HDT), above which SpyCatcher proteins are placed and connected to the HDT via linkers.” To mitigate ordering effects, half of the participants started with the traditional GUI-based modeling interface, while the other half began with the language-based Chat Modeling interface. During the study, the experimenter provided assistance when participants requested help, particularly for unfamiliar GUI operations.

To evaluate user experience and perceived task workload, we employed the System Usability Scale (SUS) [42] and the NASA Task Load Index (NASA-TLX) [43] for the Chat Modeling prototype. In addition, participants completed a questionnaire assessing Chat Modeling-specific features. Finally, we conducted semi-structured interviews with each participant to collect qualitative insights.

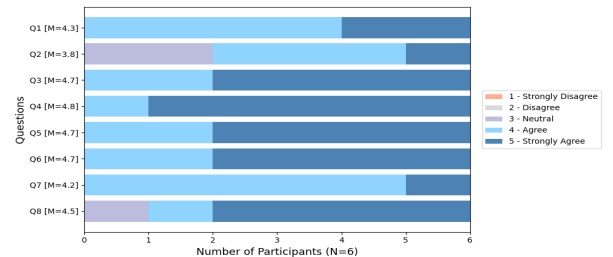


Fig. 11: Questionnaire results with a 5-point Likert scale.

10.1.2 Results

All six participants successfully completed the study and were included in the analysis. On average, participants completed the tasks in 8.6 min with Chat Modeling and 12.3 min with the GUI-based MesoCraft condition. To assess the overall user experience of the Chat Modeling prototype, we computed the SUS score, which achieved an average of **80.4** ($SD = 3.68$), indicating high perceived usability. To evaluate perceived task workload, we calculated the weighted NASA-TLX score. Although the original NASA-TLX uses 20-point scales for individual dimensions, we normalized the scores to a 100-point scale for consistency and ease of interpretation, where higher scores indicate lower perceived workload. All dimensions were assigned equal weights when computing the overall score. The average inverted NASA-TLX score was **81.5** ($SD = 7.0$), suggesting a low workload during task execution. In addition, all participants completed a custom questionnaire targeting Chat Modeling-specific features (see Table 2) using a 5-point Likert scale. The questionnaire focused on participants’ perceived usability, expressiveness, control, and

usefulness of the Chat Modeling prototype. As shown in Figure 11, the mean scores for all questions exceeded 4, except Q2, indicating generally positive user feedback toward the system.

10.1.3 Feedback

After completing the questionnaire, all participants took part in a follow-up interview consisting of two open-ended questions. Overall, participants expressed a positive attitude toward language-based modeling and found Chat Modeling to be a promising alternative to traditional GUI-driven workflows.

Several participants highlighted the advantages of procedural and language-based modeling over manual operations. In particular, P1 and P6 appreciated the ability to efficiently replicate instances without repetitive manual manipulation, noting that the system “allows rapid duplication of instances instead of adjusting each one individually.” P2, P3, and P4 emphasized that natural language interaction lowers the barrier to entry by reducing reliance on detailed GUI knowledge. They noted that even users with little to no prior experience in 3D modeling could quickly begin constructing models using Chat Modeling. P5 noted that an LLM-driven interface is particularly beneficial for providing domain knowledge and assisting with high-level planning during biological modeling tasks.

At the same time, participants pointed out that Chat Modeling still entails a learning curve, especially for users with no prior exposure to MesoCraft. Several participants remarked that a basic understanding of the underlying modeling concepts and system structure is still required, which helps explain why Q2 did not exceed an average score of 4 in the questionnaire.

Participants also identified several missing features in the current system. P2 and P3 pointed out that the system currently lacks support for fine-grained modifications of individual proteins, such as editing specific residues or adjusting detailed visual properties (e.g., changing the color of a particular amino acid). P1 observed that novice users often issue ambiguous instructions during the planning stage due to limited modeling experience, and suggested that the system could improve usability by proactively prompting users to clarify underspecified commands. Finally, P6 noted that the current system relies on pre-existing MesoCraft skeletons and proteins, and suggested that greater automation, such as automatically querying and downloading relevant PDB files and skeletons, would further enhance the system’s usability and scalability.

10.2 Expert Interview

In this section, we detail the procedure and outcomes of expert evaluations conducted to validate the efficiency of the prototype.

10.2.1 Procedure

To assess the Chat Modeling prototype, we continued our engagement with the three experts initially interviewed in section 3. This choice was driven by multiple considerations. Firstly, their insights are invaluable for verifying whether the prototype meets the expectations set during the formative study and for reflecting on the design requirements. Secondly, their extensive experience in biological structure modeling provides a rare and essential perspective for assessing the prototype against real use cases. Thirdly, their experience guiding new users of MesoCraft equips them with a thorough understanding of the challenges novices face, further enriching their assessment of our prototype.

Our evaluations took the form of semi-structured interviews, with each expert being interviewed individually. The semi-structured format was chosen for its balance of focus and flexibility. This approach enables the investigator to maintain direction on predefined questions while also having the freedom to delve into emerging ideas [44]. Each session lasted approximately 60 to 90 minutes. Initially, we introduced the prototype to the experts, allowing them a brief overview. We then demonstrate our new modeling workflow from uploading a paper to a final 3D models. Then the experts are invited to freely explore the prototype. This hands-on experience enabled them to provide feedback on the prototype’s features and functionalities. Consistent with the nature of semi-structured interviews, the facilitator concluded each session by posing additional questions not previously covered during the interview, ensuring a comprehensive collection of insights and reflections. The detailed semi-structured questions are listed in the supplemental material.

10.2.2 Outcome

From the semi-structured interviews, we extracted critical findings about different aspects. These insights, highlighting the application’s strengths and areas for improvement, have been instrumental in identifying helpful or missing features. Below, we outline these findings, which collectively inform the direction of future work.

Modes: We summarize feedback from our experts on the two modeling modes, *i.e.*, planning mode and step-by-step mode. Regarding the planning mode, all experts were impressed with this innovative approach to 3D modeling. Nonetheless, they noted that it remains conceptual and requires further improvement in future work. E2 observed, “It is impressive that it can indicate which of its statements are drawn from the uploaded paper, as this greatly enhances credibility.” E3 remarked, “This multimodal input approach expands the possibilities of biological modeling, as the field is characterized by a vast body of literature. A promising future direction is to incorporate multiple papers as input, rather than relying on a single paper as is currently the case.” For the step-by-step mode, E1 noted, “This mode is well-suited for users who already know the modeling order and steps of modeling but are unsure how to operate the software. Compared with the planning mode, it also saves the time required for multi-turn dialogues to establish a plan.”

Usability: All experts praised the usability of Chat Modeling, affirming its applicability within their workflows for modeling biological structures. Notably, E1 emphasized the convenience and precision of using natural language for position updates, stating, “This textual way is more convenient than mouse dragging and offers greater precision because I can specify the residue index.” E3 remarked, “The Chat Modeling tool can serve as a copilot, where users can gather inspiration when uncertain about the exact operations.” We inquired about their opinions on the current response latency, to which they unanimously found it to be entirely acceptable. E1 and E2 highlighted the value of providing initial introductory sentences for new users, commenting, “The current introductory sentences effectively inform users about how to operate the system. However, it would be beneficial to shorten these introductions once the user is familiar with the system.”

Interactivity: Given that Chat Modeling introduces new interactive design, a focal point of our semi-structured interviews was the opinions of the new interactivity. Regarding the intent-conditioned widgets features, all three experts concurred that “the combination of mouse clicking and natural language interaction

offers a superior modeling experience.” Furthermore, one expert (E2) remarked, “Relying solely on natural language interactions may be burdensome for users, especially for actions like applying rules. Therefore, incorporating buttons within the chat box could markedly enhance modeling efficiency.” E1 emphasized the importance of the selection button’s design, noting, “Users may not always know the exact names of the ingredients due to the presence of variants and complex terminologies. Hence, offering a selection of similar ingredients is a thoughtful design choice. Nonetheless, providing a 3D preview of each ingredient’s structure would greatly enrich the user experience.” For the interactive plan execution feature, E1 noted, “This interactive execution provides room to adjust the model at each step, since the planning phase does not allow users to see the intermediate results of execution. With interactive execution, I can make real-time adjustments.” E3 mentioned, “this interactive execution also serves as a complement to the executed plan, as it allows users to introduce new modeling operations during execution. Before proceeding to the next step, users can still provide additional instructions.”

Features: We inquired with our experts about the aspects of the Chat Modeling prototype they found most valuable, as well as any features or capabilities they felt were missing. All three experts acknowledged the innovation of the modeling memory and its potential to enhance the user experience. E1 suggested, “Providing feedback through dialogue when I am dissatisfied with an execution result is a valuable feature, as it offers a channel for user feedback. However, the current system requires me to click a button before giving feedback. A more seamless design would enable the system to automatically distinguish whether an instruction is feedback or a modeling operation.” E3 expressed approval of the high-level parameter design, observing, “The current parameters in MesoCraft primarily consist of quantitative variables, which do not consider the skeleton’s or other elements’ properties. Incorporating high-level parameters allows the modeling to reflect real-world properties more accurately, enhancing scientific correctness.” E2 noted that the ability to reuse previous modeling plans and enable knowledge transfer is highly valuable, as the biological domain contains numerous similar structures. This functionality facilitates the rapid modeling of related structures. Regarding missing features or commands, our experts identified several enhancements that could further improve the tool’s functionality. Firstly, they suggested making modifications to visual representations more detailed, such as enabling color adjustments at the residue level. Secondly, they recommended introducing a feature that allows users to upload local ingredients or download ingredients from the internet during their conversation. Lastly, they valued the system’s ability to interpret instructions like “populate more elements,” turning qualitative statements into quantifiable actions.

Overall, our three experts were pleasantly surprised about the prototype and excited about its possibilities for 3D biological structure modeling.

11 DISCUSSION

Biological modeling vs. general 3D modeling: Prior autonomous modeling in computer graphics [45], [46], [47], and more recent text-to-3D generators, primarily pursue perceptual plausibility for everyday objects; in many graphics settings, whether the output is exact is secondary to whether it looks reasonable. Biological visualization has a fundamentally different goal. Here, many

components (e.g., proteins) have fixed, experimentally determined shapes, and the task is to compose them under strict geometric and quantitative constraints derived from literature and data. As a result, models must be tightly controlled, auditable, and reproducible, not sampled as “random but plausible” content. This distinction motivates our approach: rather than free-form shape synthesis, we emphasize literature-grounded planning and controllable composition of known components, ensuring outputs are scientifically valid, not merely visually convincing.

Beyond biological modeling, we see the most direct applicability of our design in other constraint-driven modeling and visualization tasks where users assemble known components under domain-specific rules, rather than synthesize arbitrary shapes. Examples include mechanical assembly visualization, materials modeling, and educational simulations of physical systems. In these settings, intent-level control, inspectable plans, and interactive execution can help users refine high-level goals into executable operations while preserving domain constraints. However, the current implementation of Chat Modeling is still specialized for biological modeling: its agents, memory contents, widgets, and execution interface are tied to biological modeling rules and the MesoCraft environment. Applying the framework to other domains would therefore require redefining the domain constraints, replacing the modeling backend, and designing task-specific widgets and validation mechanisms.

Failure Cases and Validation: During the qualitative and expert evaluation, we identified several failure cases that our method cannot address. The main types of these failure cases include mistakenly identifying certain biological concepts, such as cell walls, as skeletons; failing to identify rule parameters during a rule creation operation, such as the parameter specifying “populate thirty spike proteins on the membrane surface”; and being unable to identify rule types accurately. Additionally, there are operations that our system currently cannot support, which should be addressed in future work. These include operations like “populate more” degree representation and “Place the connexins within the membrane gap.” The latter task requires precise geometry, spatial information, and collision-handling algorithms to position the elements accurately.

Limitations: Our system represents an early-stage prototype and makes several simplifying design choices. First, it supports interactive refinement during execution but does not yet automatically revise the remaining plan steps after user edits. This static-plan assumption is reasonable for local refinements, such as parameter tuning or minor visual adjustments, but may break down when users skip, reorder, defer, or fundamentally alter steps, since such changes can affect downstream dependencies. Future work should explore partial replanning to distinguish local edits from structural plan changes. Second, our current prototype primarily relies on user inspection and user-reported corrections to identify such errors. This mechanism is useful for capturing explicit feedback, but it does not automatically detect inconsistencies between agents. This places part of the error-detection burden on users, who may not always recognize subtle inter-agent inconsistencies. A promising future direction is to introduce automated validation across agents, such as verifying parameter ranges against domain constraints or rendering-based checks to flag suspicious outputs. Finally, our memory update strategy is based on interaction counts rather than continuous or time-based adaptation, which we leave for future work.

12 CONCLUSION

In this paper, we investigated the potential of the LLM-powered agent framework to enhance scientific visualization and 3D modeling tasks. We introduced a literature-grounded agent framework that assists model planning and execution. Additionally, we propose several interaction designs for plan execution that extend text-only natural language interaction. We also introduce a customized modeling memory for long-term refinement and reusability. Through quantitative evaluation, an exploratory user study, and expert interviews, we gathered insights that underscore the framework's promise for 3D biological modeling. The feedback highlighted the framework's effectiveness and potential for broader application in scientific modeling.

ACKNOWLEDGMENTS

The research was supported by the King Abdullah University of Science and Technology (BAS/1/1680-01-01).

REFERENCES

- [1] D. S. Goodsell, A. J. Olson, and S. Forli, "Art and science of the cellular mesoscale," *Trends in Biochemical Sciences*, vol. 45, no. 6, pp. 472–483, 2020.
- [2] Autodesk, Inc., "Maya - 3D Computer Animation, Modeling, Simulation, and Rendering Software," <https://www.autodesk.com/products/maya/overview>, 2026, accessed: 2026-01-01.
- [3] Blender Foundation, "Blender - a 3D modelling and rendering package," <https://www.blender.org/>, 2026, accessed: 2026-01-01.
- [4] B. Kozlíková, M. Krone, M. Falk, N. Lindow, M. Baaden, D. Baum, I. Viola, J. Parulek, and H.-C. Hege, "Visualization of biomolecular structures: State of the art revisited," *Computer Graphics Forum*, vol. 36, no. 8, pp. 178–204, 2017. [Online]. Available: <https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.13072>
- [5] D. Surís, S. Menon, and C. Vondrick, "Vipergpt: Visual inference via python execution for reasoning," *arXiv preprint arXiv:2303.08128*, 2023.
- [6] T. Gupta and A. Kembhavi, "Visual programming: Compositional visual reasoning without training," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 14 953–14 962.
- [7] O. Michel, A. Bhattad, E. VanderBilt, R. Krishna, A. Kembhavi, and T. Gupta, "Object 3dit: Language-guided 3d-aware image editing," *Advances in Neural Information Processing Systems*, vol. 36, 2024.
- [8] W. Huang, C. Wang, R. Zhang, Y. Li, J. Wu, and L. Fei-Fei, "Voxposer: Composable 3d value maps for robotic manipulation with language models," *arXiv preprint arXiv:2307.05973*, 2023.
- [9] Y.-J. Wang, B. Zhang, J. Chen, and K. Sreenath, "Prompt a robot to walk with large language models," *arXiv preprint arXiv:2309.09969*, 2023.
- [10] N. Mohammad Khalid, T. Xie, E. Belilovsky, and T. Popa, "Clip-mesh: Generating textured meshes from text using pretrained image-text models," in *SIGGRAPH Asia 2022 Conference Papers*, ser. SA '22. ACM, Nov. 2022. [Online]. Available: <http://dx.doi.org/10.1145/3550469.3555392>
- [11] C. Sun, J. Han, W. Deng, X. Wang, Z. Qin, and S. Gould, "3d-gpt: Procedural 3d modeling with large language models," *arXiv preprint arXiv:2310.12945*, 2023.
- [12] Y. Ma, X. Zhang, X. Sun, J. Ji, H. Wang, G. Jiang, W. Zhuang, and R. Ji, "X-mesh: Towards fast and accurate text-driven 3d stylization via dynamic textual guidance," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2023, pp. 2749–2760.
- [13] W. Feng, W. Zhu, T.-j. Fu, V. Jampani, A. Akula, X. He, S. Basu, X. E. Wang, and W. Y. Wang, "Layoutgpt: Compositional visual planning and generation with large language models," *Advances in Neural Information Processing Systems*, vol. 36, 2024.
- [14] Y. Zhou, A. I. Muresanu, Z. Han, K. Paster, S. Pitis, H. Chan, and J. Ba, "Large language models are human-level prompt engineers," *arXiv preprint arXiv:2211.01910*, 2022.
- [15] A. Narechania, A. Srinivasan, and J. Stasko, "NL4DV: A toolkit for generating analytic specifications for data visualization from natural language queries," *IEEE Transactions on Visualization and Computer Graphics*, vol. 27, no. 2, pp. 369–379, 2020.
- [16] R. Mitra, A. Narechania, A. Endert, and J. Stasko, "Facilitating conversational interaction in natural language interfaces for visualization," in *2022 IEEE Visualization and Visual Analytics (VIS)*. IEEE, 2022, pp. 6–10.
- [17] Y. Wang, Z. Hou, L. Shen, T. Wu, J. Wang, H. Huang, H. Zhang, and D. Zhang, "Towards natural language-based visualization authoring," *IEEE Transactions on Visualization and Computer Graphics*, vol. 29, no. 1, pp. 1222–1232, 2022.
- [18] P. Maddigan and T. Susnjak, "Chat2VIS: Fine-Tuning Data Visualisations using Multilingual Natural Language Text and Pre-Trained Large Language Models," *arXiv preprint arXiv:2303.14292*, 2023.
- [19] D. Jia, A. Irger, L. Besancon, O. Strnad, D. Luo, J. Bjorklund, A. Kouyoumdjian, A. Ynnerman, and I. Viola, "VOICE: Visual Oracle for Interaction, Conversation, and Explanation," vol. 31, no. 10, pp. 8828–8845, Oct. 2025. [Online]. Available: <https://doi.ieeecomputersociety.org/10.1109/TVCG.2025.3579956>
- [20] A. Raistrick, L. Lipson, Z. Ma, L. Mei, M. Wang, Y. Zuo, K. Kayan, H. Wen, B. Han, Y. Wang *et al.*, "Infinite photorealistic worlds using procedural generation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 12 630–12 641.
- [21] F. De La Torre, C. M. Fang, H. Huang, A. Banburski-Fahey, J. Amores Fernandez, and J. Lanier, "Llmr: Real-time prompting of interactive worlds using large language models," in *Proceedings of the CHI Conference on Human Factors in Computing Systems*, 2024, pp. 1–22.
- [22] P. Prusinkiewicz and A. Lindenmayer, *The algorithmic beauty of plants*. Springer Science & Business Media, 2012.
- [23] J. Guo, H. Jiang, B. Benes, O. Deussen, X. Zhang, D. Lischinski, and H. Huang, "Inverse procedural modeling of branching structures by inferring l-systems," *ACM Transactions on Graphics (TOG)*, vol. 39, no. 5, pp. 1–13, 2020.
- [24] T. Niese, S. Pirk, M. Albrecht, B. Benes, and O. Deussen, "Procedural urban forestry," *ACM Transactions on Graphics (TOG)*, vol. 41, no. 2, pp. 1–18, 2022.
- [25] A. Webanck, Y. Cortial, E. Guérin, and E. Galin, "Procedural cloudscapes," in *Computer Graphics Forum*, vol. 37, no. 2. Wiley Online Library, 2018, pp. 431–442.
- [26] E. Galin, A. Peytavie, N. Maréchal, and E. Guérin, "Procedural generation of roads," in *Computer Graphics Forum*, vol. 29, no. 2. Wiley Online Library, 2010, pp. 429–438.
- [27] Y. I. Parish and P. Müller, "Procedural modeling of cities," in *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, 2001, pp. 301–308.
- [28] P. Müller, P. Wonka, S. Haegler, A. Ulmer, and L. Van Gool, "Procedural modeling of buildings," in *ACM SIGGRAPH 2006 Papers*, 2006, pp. 614–623.
- [29] M. Schwarz and P. Müller, "Advanced procedural modeling of architecture," *ACM Transactions on Graphics (TOG)*, vol. 34, no. 4, pp. 1–12, 2015.
- [30] N. Nguyen, O. Strnad, T. Klein, D. Luo, R. Alharbi, P. Wonka, M. Maritan, P. Minde, L. Autin, D. S. Goodsell *et al.*, "Modeling in the time of COVID-19: Statistical and rule-based mesoscale models," *IEEE transactions on visualization and computer graphics*, vol. 27, no. 2, pp. 722–732, 2020.
- [31] J. Winfer, A. Syed, P. Ekers, L. Thistle, N. Nguyen, O. Strnad, D. Goodsell, I. Viola, and D. Luo, "T4 model," <https://www.nanovis.org/T4-model.html>, accessed on 01/01/2026.
- [32] M. Z. Baig, O. Strnad, I. Viola, and D. Luo, "Chloroplast Model," <https://www.nanovis.org/Chloroplast-model.html>, accessed on 01/01/2026.
- [33] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.
- [34] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. D. O. Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman *et al.*, "Evaluating large language models trained on code," *arXiv preprint arXiv:2107.03374*, 2021.
- [35] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W. tau Yih, T. Rocktäschel, S. Riedel, and D. Kiela, "Retrieval-augmented generation for knowledge-intensive nlp tasks," 2021. [Online]. Available: <https://arxiv.org/abs/2005.11401>
- [36] Z. Huang, S. Gutierrez, H. Kamana, and S. MacNeil, "Memory sandbox: Transparent and interactive memory management for conversational agents," in *Adjunct Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, 2023, pp. 1–3.
- [37] W. Zhong, L. Guo, Q. Gao, H. Ye, and Y. Wang, "Memorybank: Enhancing large language models with long-term memory," in *Proceedings of*

- the AAAI Conference on Artificial Intelligence*, vol. 38, no. 17, 2024, pp. 19 724–19 731.
- [38] H. Yao, Y. Song, Y. Chen, N. Wu, J. Xu, C. Sun, J. Zhang, T. Weng, Z. Zhang, Z. Wu *et al.*, “Molecular architecture of the sars-cov-2 virus,” *Cell*, vol. 183, no. 3, pp. 730–738, 2020.
 - [39] K. Guo, R. Grünberg, Y. Ren, T. Chang, S. Wustoni, O. Strnad, A. Koklu, E. Díaz-Galicia, J. P. Agudelo, V. Druet *et al.*, “SpyDirect: A Novel Bio-functionalization Method for High Stability and Longevity of Electronic Biosensors,” *Advanced Science*, p. 2306716.
 - [40] OpenAI, “Hello gpt-4o,” 2024, accessed: 2026-01-01. [Online]. Available: <https://openai.com/index/hello-gpt-4o/>
 - [41] —, “GPT-4 Technical Report,” 2024.
 - [42] A. Bangor, P. T. Kortum, and J. T. Miller, “An empirical evaluation of the system usability scale,” *Intl. Journal of Human–Computer Interaction*, vol. 24, no. 6, pp. 574–594, 2008.
 - [43] S. Hart, “Development of nasa-tlx (task load index): Results of empirical and theoretical research,” *Human mental workload/Elsevier*, 1988.
 - [44] O. A. Adeoye-Olatunde and N. L. Olenik, “Research and scholarly methods: Semi-structured interviews,” *Journal of the american college of clinical pharmacy*, vol. 4, no. 10, pp. 1358–1367, 2021.
 - [45] A. Kasper, Z. Xue, and R. Dillmann, “The kit object models database: An object model database for object recognition, localization and manipulation in service robotics,” *The International Journal of Robotics Research*, vol. 31, no. 8, pp. 927–934, 2012.
 - [46] S. Khalfaoui, R. Seulin, Y. Fougerolle, and D. Fofi, “An efficient method for fully automatic 3D digitization of unknown objects,” *Computers in Industry*, vol. 64, no. 9, pp. 1152–1160, 2013.
 - [47] S. Kriegel, “Autonomous 3D modeling of unknown objects for active scene exploration,” Ph.D. dissertation, Technische Universität München (TUM), 2015.