

A Hybrid GUI-LLM Interface Paradigm for 3D Scene Customisation

Isaac Valadez
WISE Lab
Vrije Universiteit Brussel, Pleinlaan 2
Brussels, Belgium
jvaladez@vub.be

Irving Avina
Mad Monkeys
Ottawa, Canada
irving@madmonkeys.io

Beat Signer
WISE Lab
Vrije Universiteit Brussel, Pleinlaan 2
Brussels, Belgium
bsigner@vub.be

Abstract

3D environments offer powerful capabilities for simulating and monitoring complex systems. However, they remain inaccessible to non-expert users who lack specialised training. We present the Layered Customisation System (LCS), a hybrid interface that combines GUI-based direct manipulation with LLM-powered natural language interaction for customising 3D scenes. Through a controlled study with 12 participants, each completing three sessions with different interfaces (GUI-only, LLM-only and hybrid), we investigated how interaction types affect performance, errors and user preferences. Key findings include a reduced learning effect across sessions for users starting with the LLM interface and a trend where participants who first experienced GUI-based interaction showed higher rates of recognising LLM errors later. We contribute empirical evidence of GUI-LLM modality trade-offs in 3D interaction, the selector-layer architecture for hybrid interfaces, and design recommendations, including GUI-first onboarding, enabling prompts as discovery tools and supporting prompt-then-refine workflows.

CCS Concepts

• **Human-centred computing** → **HCI design and evaluation methods**; **Natural language interfaces**; *Graphical user interfaces*; *Empirical studies in HCI*.

Keywords

3D interaction, digital twins, natural language interfaces, hybrid interaction, visual customisation, large language models

ACM Reference Format:

Isaac Valadez, Irving Avina, and Beat Signer. 2026. A Hybrid GUI-LLM Interface Paradigm for 3D Scene Customisation. In *Designing Interactive Systems Conference (DIS '26)*, June 13–17, 2026, Singapore, Singapore. ACM, New York, NY, USA, 15 pages. <https://doi.org/10.1145/3800645.3812986>

1 Introduction

3D environments often require users to be familiar with spatial navigation metaphors [12] and specialised training. This can be the case for professionals like urban planners who interact with 3D city-scale replicas and filter buildings by various attributes (e.g. type, state or energy efficiency) to highlight patterns and create custom views that communicate insights to different audiences. Another

case is that of regular citizens who are introduced to Digital Twins in citizen-centric planning exercises. In these cases, the usefulness of these replicas or “scenes” is limited by their steep learning curve and general-purpose interfaces (as opposed to task-oriented interfaces). Additionally, most existing software primarily caters to technical experts, offering advanced but not user-centric interfaces. As a result, non-expert stakeholders often feel overwhelmed by the complexity of 3D environments.

Existing attempts for end-user customisation are often limited to high-level configuration tools or programming scripts, which pose major barriers to anyone lacking programming expertise and vendor-specific knowledge.

We propose the *Layered Customisation System (LCS)*, an interface paradigm that addresses these challenges through three key features:

- (1) *Semantic Selector-Layer Architecture*. LCS introduces a declarative model where users define *what* they want to customise and specify it using *selectors*. Selectors are combined with *layers* (named after a common interaction pattern found in design software) that specify *how* users want to customise their selection. Selectors use semantic attributes to identify groups of objects (e.g. “*all buildings with height > 50m*”), while layers specify visual transformations (e.g. “*colour: red*”, “*opacity: 0.5*”). This separation enables users to think at a semantic level while maintaining precise control over visual output. The system maintains these selector-layer groups (called composites) as persistent, editable rules that can be reordered, combined and refined iteratively.
- (2) *Hybrid GUI-LLM Interaction*. LCS integrates GUI and natural language interaction. This hybrid approach takes advantage of the strengths of each interface type: LLM prompts for rapid, high-level specification and GUI for direct manipulation and refinement.
- (3) *Developer-friendly Framework*. Beyond the end-user interface, LCS provides a flexible framework that enables developers to create their own customisation interfaces for different domains and use cases.

In our study, we tested the Layered Customisation System in the context of a city’s digital twin. As a result, most examples are based on objects commonly found in public spaces within cities. However, our architecture is flexible and works with all types of 3D scenes. A 3D scene is a computer-generated three-dimensional environment (e.g. a city model or building interior) where users can navigate, inspect and interact with virtual objects.

Our work investigates how the presented hybrid GUI-LLM interface paradigm affects user performance, behaviour and preferences



This work is licensed under a Creative Commons Attribution 4.0 International License. *DIS '26, Singapore, Singapore*

© 2026 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-2563-0/2026/06
<https://doi.org/10.1145/3800645.3812986>

when interacting with 3D environments. In particular, we address the following *four research questions*:

- *RQ1*. How do different interface types affect task performance? We examine how GUI-only, LLM-only and hybrid GUI-LLM interaction modalities influence performance metrics, including task completion time and error frequency across different types of 3D customisation tasks.
- *RQ2*. Do interface types influence error patterns and user awareness? We investigate whether different interaction types lead users to make more errors and identify errors made by the system.
- *RQ3*. How do users evaluate the usability of the three different interfaces? We assess user opinions for each interaction modality with respect to control, efficiency and personal preference.
- *RQ4*. How does the initially introduced interface type shape subsequent task performance? We examine whether the first encountered interface type affects users' error rates and learning trajectories across the rest of the study.

With the presented research, we make three primary contributions to the HCI community:

- (1) *LCS Framework for 3D Scene Customisation*. We present an architectural pattern for 3D visualisation interfaces based on semantic selectors and visual layers. We detail the system architecture, implementation considerations and demonstrate its applicability across different use cases. The framework is designed to be extensible, supporting integration with various 3D engines, natural language models and interface paradigms.
- (2) *Empirical Evidence of GUI-LLM Modality Trade-offs in 3D Interaction*. Through a controlled study with 12 participants each completing 3 sessions with a different interface type, we provide preliminary quantitative evidence of how interaction types might affect performance in 3D customisation tasks. Our findings suggest an efficiency-learning trade-off, where the interface with the fastest average task completion time provided fewer learning opportunities, and an exploratory trend suggesting that initial GUI exposure may enhance users' ability to recognise when an LLM performs actions in a way different to what the users expected.
- (3) *Design Recommendations for Hybrid 3D Interfaces*. Based on our empirical findings, we contribute with design recommendations for hybrid GUI-LLM interfaces.

Our work demonstrates that hybrid GUI-LLM interfaces represent a promising direction to easing the customisation and personalisation of complex 3D visualisations by non-technical users. We start in Section 2 by briefly discussing research that has been conducted in the Human-Computer Interaction field on digital 3D environments, as well as LLM and hybrid interfaces for various purposes. We continue in Section 3 by providing an overview of our system's goals and design, followed by an explanation of our study in Section 4, its results in Section 5 and our interpretation of those results in Section 6. After discussing the limitations of the presented approach in Section 7, we conclude by outlining some future work in Section 8.

2 Related Work

Our work builds upon three main research areas: interaction techniques for 3D environments, natural language interfaces for spatial data and hybrid interaction paradigms. We reviewed each area to identify gaps that our approach addresses and position our contributions within the broader HCI landscape.

2.1 3D Environment Interaction

The challenge of interacting with 3D environments has been extensively studied in the HCI community. Jankowski and Hachet [12] provide a comprehensive survey identifying fundamental interaction tasks in 3D spaces, such as navigation, selection, manipulation and system control. However, these traditional paradigms assume users possess spatial reasoning skills and are familiar with 3D manipulation metaphors, which is often not the case.

Besançon et al. [2] presented a systematic review of spatial interfaces for 3D visualisation, revealing a fragmented landscape of interaction techniques. They classified approaches into tactile interaction (touch screens and tablets), tangible interaction (physical props), mid-air gestures and hybrid combinations. While each paradigm offers benefits (e.g. tactile for precision, tangible for naturalness or gestures for expressiveness), they noted that “*experts in different domains have different needs*” and highlighted the lack of adoption of spatial 3D visualisation techniques by domain experts. This adoption gap stems partly from the cognitive overhead of mastering these interfaces.

Direct manipulation interfaces, while intuitive for simple tasks, struggle with complex selections in dense 3D scenes. Jankowski and Hachet [12] identified the tension between WIMP (Windows, Icons, Menus, Pointer) interfaces that split functionality between 2D GUI and 3D visualisation, versus post-WIMP approaches that embed controls directly in 3D space. Both approaches force users to translate domain-level intentions (e.g. “*highlight inefficient buildings*”) into low-level geometric operations (e.g. selecting individual polygons).

Several systems have attempted to bridge this gap through specialised interfaces. For Building Information Modelling (BIM), researchers have developed domain-specific tools that provide semantic filters and predefined views. However, these solutions remain application-specific and lack flexibility for new queries or customisations. Game interfaces demonstrate progressive disclosure through what Shneiderman described as a “*layered or spiral approach to learning*”, where complexity increases with user expertise [22]. World of Warcraft exemplifies this approach: beginner players see only essential UI components, with functionality gradually revealed as they gain experience. Yet translating these progressive disclosure concepts to professional 3D visualisation tools remains challenging, as domain experts need immediate access to advanced features while maintaining ease of use. In our work, rather than requiring users to adapt to 3D interaction paradigms, we enable them to express intentions at the semantic level using familiar GUI controls and natural language.

3D Digital Twins serve as good examples of the previously mentioned challenges. They tend to provide either overwhelming details that obscure important patterns or oversimplified dashboards that lack spatial context [8]. They offer powerful capabilities, but require

extensive training and lack user involvement during system design and evaluation [1].

On a different aspect, the semantic interoperability challenge in digital twins is particularly relevant to our work. Inokuchi et al. [11] proposed semantic digital twin architectures that “leverage semantics to construct a designer-oriented digital twin”. They argue for metadata schemas that enable generic descriptions of entities’ dynamic behaviours. This semantic layer aligns with our selector-based approach, which operates on object attributes rather than raw geometry.

Our work addresses these challenges by providing an accessible interface layer above complex 3D scenes. Rather than replacing existing platforms, we complement them with customisation capabilities.

2.2 Natural Language Interfaces for Spatial Data

Natural language processing (NLP) has emerged as a powerful paradigm for making complex systems more accessible. In the visualisation domain, natural language interfaces promise to lower barriers by allowing users to express queries in their own terms rather than learning specialised syntax or interfaces. Shen et al. [21] provided a comprehensive survey of natural language interfaces for data visualisation, reviewing systems from Eviza to NL4DV using the information visualisation pipeline framework. Their analysis reveals both the promise and limitations of Visual Natural Language Interfaces (V-NLIs) for democratising data exploration. NL4DV demonstrates how NL queries can be translated to Vega-Lite specifications for chart generation [16], while FlowSense enables users to construct complex visualisation pipelines using plain English within a dataflow system [27]. ChartGPT specifically addresses the task of transforming abstract or ambiguous natural language inputs into chart specifications [23]. However, research also shows that users need to have clear goals and be able to articulate them through high-quality prompts [19]. This observation is particularly relevant for 3D visualisation, where spatial relationships and visual properties might be difficult to articulate precisely in text.

Planas et al. [18] propose a model-driven approach for AI-based user interfaces, arguing that conversational interfaces (CUIs) can complement traditional GUIs by enabling more natural interactions. They identify a critical challenge: CUIs are often “created as standalone components using platform-dependent libraries and technologies”. This fragmentation makes it difficult to integrate NLP capabilities into existing visualisation systems. In other cases, NLP and LLMs have been used to enable interaction with existing GUIs through conversational means [24].

Other systems have explored natural language for 2D data visualisation. Tools like DataFormulator [25] demonstrate how NLP can facilitate iterative chart creation by combining natural language with direct manipulation. Like us, they identify the potential of free-form text prompts to provide *unbounded expressiveness*, but also point out that they miss the precision and affordance of UI interactions. This tension between expressiveness and precision becomes even more pronounced in 3D environments, where users often struggle to formulate spatial relationships.

2.3 Hybrid Interaction

Combining interaction modalities for spatial tasks has long been studied. As early as 1980, Bolt’s ‘Put That There’ [4] demonstrated a system in which users could create and move geometric shapes on a display by combining spoken commands (e.g. “put that there”) with deictic pointing gestures. In his work, speech indicated the operation to perform while gestures indicated to *which* object and *where* to place it. Oviatt [17] later examined how users combined speech and pen input when interacting with map systems. In her studies, she identified that users distributed information across speech and pen input, using language for descriptive content (e.g. identifying a type of object) and spatial input for location and form. She further found that this distribution reduced errors compared to speech-only interaction.

Hybrid GUI-LLM interfaces are more text-mediated than Bolt’s and Oviatt’s speech-and-gesture systems. Still, they follow a similar *command-and-control* paradigm that operates at two abstraction levels: expressing intent in natural language and specifying explicit parameters via GUI controls. These types of hybrid interfaces are particularly relevant for complex tasks that can benefit from both rapid specification and precise control. In their study, Cao et al. [6] highlighted several benefits of hybrid interfaces (GUI + LLM prompts), including system transparency, continuity throughout task completion and traceability of actions. In other studies, researchers observed that in hybrid interfaces, users tend to use AI to speed up data analysis, whereas direct manipulation can be beneficial for users who make continuous adjustments [14].

Low-code/no-code platforms offer another perspective on hybrid interaction. LowCoder [19] specifically combines visual block-based programming with natural language. The team found that “*language models helped users discover previously unknown operators in 75% of tasks*” compared to 22.5% with traditional search. However, they also note that NLP did not support novices when they were unsure about how to complete a task.

Recent work has begun exploring hybrid GUI-LLM interfaces more directly. Wang et al. [24] investigated the use of LLMs to enable conversational interactions with mobile UIs, addressing modality switching between visual GUIs and natural language. ReactGenie [26] demonstrates how touch, voice and GUI interactions can be combined using LLMs to compose multimodal commands executing operations that normally require multiple GUI interactions. Stylette [13] enables users to change visual design by clicking components and expressing goals in natural language, achieving 35% faster task completion versus GUI-only tools. For 3D specifically, GesPrompt [10] combines speech with co-speech gestures for LLM-based XR interaction, while research on modality switching reveals key factors affecting touch versus voice preferences [7].

3 Hybrid GUI-LLM System Design

Our initial goal was to create a system that allows users to move between interaction paradigms while maintaining a consistent interface. As our research progressed, several considerations shaped our approach. First, we were interested in studying how users customised complex 3D scenes to reveal the relationship between the entities present in them. We chose a digital twin of a city as these

“scenes” typically contain multiple objects with varying metadata. We expected most tasks to involve multiple buildings or areas of a city. Because of this, our interface and abstractions should favour high-level actions, such as identifying groups of objects that share semantic attributes and applying collective changes, rather than customisations at an individual level.

Further, we drew from the interaction model of layered, non-destructive editing found in design tools such as Adobe Photoshop and 3D applications like Autodesk 3ds Max, where visual operations are stacked sequentially and can be added, removed and reordered while the underlying model remains intact. This sequential, top-to-bottom execution model provides a predictable mental model where users can reason about why certain visual results appear by reading their operations in order. We also considered a node-based (dataflow graph) approach, as used in tools like Grasshopper or Unreal Engine Blueprints, which offers expressive power through branching and non-linear data flows. However, we were concerned that, as shown in previous studies [9], this type of interface would entail high cognitive costs, increase working memory demands and add difficulties for non-expert users in determining execution flows. Instead, we opted for a linear, composable model that avoids requiring users to arrange nodes or commit to a graph layout.

We also considered an interface in which LLM-generated output would be rendered as editable natural language sentences with inline controls (e.g. dropdown menus for object types or colour swatches for visual properties). While being promising, this approach introduced cascading interaction design challenges; for instance, changing an operation verb would invalidate downstream parameters. These considerations were formalised into the design goals and core concepts described in the following sections.

3.1 Design Goals

We designed our system around five principles that address the challenges of making 3D scenes accessible to non-expert users while maintaining the power needed for professional analysis:

- *Modularity.* We adopted a composable approach where small, focused operations combine to create complex visualisations. For instance, transforming schools into cubes, colouring them by floor count, and arranging them in a sorted line requires three independent layers that work together. This “atomic” approach allows users to iteratively build and experiment with custom views by mixing and matching actions.
- *Flexibility.* Layers operate on any attribute type (string, integer or float) without hard-coded dependencies. This means metadata can be mapped to different visual properties such as colour, size, shape or even dynamic properties, regardless of its original type.
- *Symmetry.* Users can achieve the same functionality through the GUI and the LLM-powered interface. Any property that can be adjusted in the GUI can be specified in a prompt, and every result from a prompt can be built manually.
- *Cross-paradigm.* Whenever possible, visual changes applied in one view (e.g. 3D) should be translatable into other views (e.g. 2D or text). This consistency across paradigms maintains semantic meaning. Note that although our current study

does not demonstrate this, our system design is prepared to handle such cases.

- *Simplification and Augmentation.* At its core, the system enables two fundamental operations: removing noise (e.g. simplification through filtering and hiding of data, objects or visual properties) and highlighting importance (e.g. augmentation through labels, colours and transformations). These capabilities address the information overload common in complex 3D scenes.

3.2 Core Concepts

Our architecture introduces three fundamental abstractions that enable flexible 3D customisation, as explained below.

3.2.1 Selectors. Selectors define *what* objects are affected by transformations. They operate on any object attribute or metadata, using conditional logic to create object groups. Selectors access all object properties, such as height, floors or usage type, through a unified interface. Users use conditional operators and gates to set selectors. Two example selector configurations are shown in Listing 1.

Listing 1: Example selector configurations

```
height > 50 AND usage = "commercial"
floors >= 3 OR basements > 0
```

While our prototype implements conditional selectors, the architecture supports spatial selectors (e.g. radius and area-based), individual object selection, as well as semantic selectors to leverage inferred relationships.

3.2.2 Layers. Layers define *how* selected objects are transformed. They represent visual operations that can modify aspects of objects’ appearance and create new objects linked to the original ones. Example transformations include:

- *Visual transformations.* Colour, opacity, size or shape replacement (e.g. turn buildings into cubes)
- *Spatial transformations.* Position remapping, arrangement in grids/lines or geographic displacement
- *Augmentation operations.* Add labels, connecting lines or distance indicators
- *Dynamic behaviours.* Flashing, scaling animations or conditional visibility

Layers should be attribute-agnostic. They are executed as composable functions applied in sequence. Each layer receives the current object state and produces a modified state.

3.2.3 Composites. Composites combine selectors and layers into visualisation rules. A composite can have zero selectors (affects all objects), one, or multiple selectors. Layers within a composite are executed top-to-bottom; in other words, layers created first are executed first. Composites are executed in creation order as well; those created first, unless re-ordered, are executed first, with later composites potentially overriding earlier transformations.

In summary, users create composites that are executed in order of creation, but they can be reordered if necessary. The first composite filters entities based on selectors’ conditions and “overlay layers” (i.e. applies visual changes). The next composite repeats this process, again taking into account all available entities and applying changes to its own selection. Figure 1 shows an example of such a composite

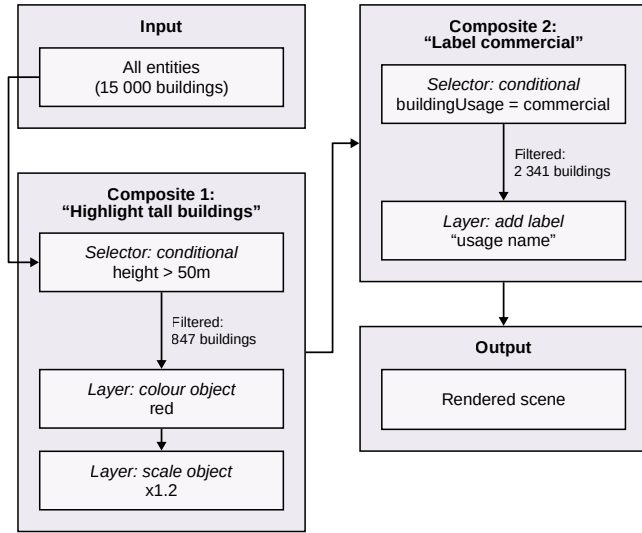


Figure 1: Example of a composite execution pipeline

execution pipeline. This hierarchical structure provides a mental model that maps naturally to how users think about visualisation tasks: “For these objects, apply these changes”.

3.3 Interface Types

3.3.1 *GUI-based Interaction.* The graphical interface provides direct manipulation through familiar controls. It has a single-column layout and consists of the following components:

- *Navigation toolbar.* Camera controls and view management
- *View tabs.* Each tab represents a different visualisation configuration, enabling rapid context switching
- *Prompt field.* Persistent text input for NLP queries, always accessible
- *Composite panels.* Collapsible panels for each composite showing selectors and layers
- *Layers and controls.* Layers and their widgets (colour pickers, sliders and checkboxes)

Text fields are used across multiple controls because they allow users to express values in different ways. For instance, users can express a desired colour in plain text or by using a hexadecimal code. In our current prototype, changes do not immediately update the 3D view due to processing constraints. Users must press the Re-execute button to update their view.

3.3.2 *LLM-based Interaction.* Large language models enable the rapid, high-level specification of complex operations. We employed Mistral Small 3.2¹ and fine-tuned it using few-shot learning on 150 example interactions. The model receives user prompts and generates structured JSON responses that our system can translate into our selectors and layers schemas. The *semantic parsing pipeline* consists of the following steps:

- (1) User prompt is sent to Mistral API with system instructions.

- (2) Model sends back JSON response specifying composites, selectors and layers.
- (3) JSON validation tool ensures structural correctness.
- (4) Parser instantiates necessary system objects (adds composites, selectors and layers to UI).
- (5) User can refresh their view.

The system successfully processes various query types, including simple filtering (e.g. “Show all buildings taller than 10 metres”), complex conditions (e.g. “schools with more than 3 floors and without basements”), and multiple operations (e.g. “Colour hospitals red and connect them to nearest schools”). Note that our current implementation does not yet include semantic inference, such as “highlight buildings taller than the city hall”.

3.3.3 *Hybrid Interaction.* The hybrid mode integrates both modalities without requiring explicit mode switching; there is always bidirectional consistency and LLM-generated operations appear identically to the ones created manually in the GUI. The only difference is that composites created via prompts retain their generating prompts, which can be edited and rerun directly in the composite GUI component.

3.4 Implementation

3.4.1 *Software architecture.* Our system’s architecture, illustrated in Figure 2, employs a client-server architecture designed to support multiple clients while maintaining a consistent data model and interaction paradigm.

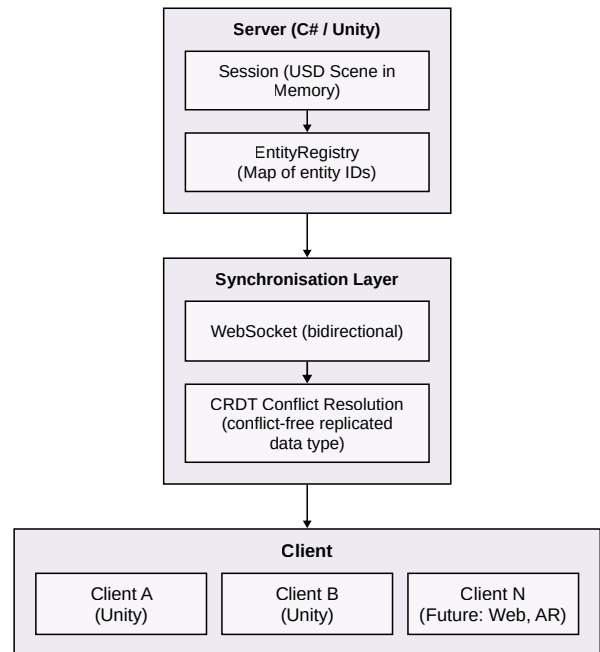


Figure 2: System architecture

We connect the server and clients via WebSockets for bidirectional, real-time communication. The server, currently implemented in C# within Unity, processes OpenUSD scene files and transforms

¹<https://docs.mistral.ai/models/mistral-small-3-2-25-06>

them into a universal Entity structure that all clients can interpret. This design anticipates future scenarios in which different clients (e.g. web browsers, AR devices or specialised visualisation tools) with different capabilities and constraints need to access the same 3D scene data. Entities are transmitted incrementally to support scenes with thousands of objects without overwhelming network bandwidth. Our prototype uses the PLATEAU dataset² from Meguro district in Tokyo, comprising approximately 15 000 building entities with rich metadata.

For collaborative scenarios, we implemented CRDT-based synchronisation, allowing multiple users to modify the same visualisation simultaneously. Composites, selectors, and layers are all synchronised in real time, enabling collaborative analysis sessions.

3.4.2 LLM setup and fine-tuning. We applied Mistral Small 3.2 (24B-Instruct-2506). A set of instructions was passed to the LLM as system instructions. These instructions included information about the available layers and their parameters, the types of selectors, entity attributes, as well as the JSON format that should be used and constraints in the type of response (i.e. not using certain symbols, respond only to topic-relevant prompts).

To fine-tune the LLM, 150 entries were provided with examples showing how to respond to different requests. Listing 2 shows such an example (symbols for correct parsing have been removed to improve readability):

Listing 2: Example used for LLM fine-tuning

```
{
  "role": "user",
  "content": "change the color of buildings to red"
}
{
  "role": "assistant",
  "content": "{ \"prompt\": \"change the color of buildings to red\", \"layers\": [
    { \"layer\": { \"name\": \"LDrawEntitiesAsGeometry\",
      \"parameters\": {} }, { \"layer\": { \"name\":
      \"LPositionByAttribute\", \"parameters\":
      { \"useLocalPosition\": true, \"positionOffset\": { \"x\": 0,
      \"y\": 0, \"z\": 0 }, \"positionScale\": \"1.0 f\" } }, { \"layer\":
      { \"name\": \"LSetGeometryColor\", \"parameters\": { \"color\":
      { \"r\": 0.6792453, \"g\": 0.1826273, \"b\": 0.1826273, \"a\": 1 },
      \"applyToChildren\": true } } }
    ], \"selectors\": { \"selector\": { \"type\": \"none\" } } }"
}
```

In this example, the LLM returns a JSON document with the original prompt, an instruction to add layers to change the colour of all objects (no selector is used in this case), to position objects in their default position and to render buildings with their default geometry. The response still passes through a validation function in C# to make sure the formatting is correct and can handle most common errors in the output, such as missing brackets. Mistral's Temperature parameter was set to zero as this is recommended for analytical tasks (this does not mean results are fully deterministic). After validation, the system processes the JSON result to generate a composite.

²https://www.mlit.go.jp/en/toshi/daisei/plateau_en_2.html

3.5 User Interface Overview

Our tool consists of two main sections. On the left-hand side, a window where the 3D objects are rendered and on the right-hand, a one-column pane where users can change their camera view, make prompts to create composites and manually create composites, as shown in Figure 3. The composite GUI component contains drop-downs to add selectors and layers, and each selector and layer has inputs that can be manually set. When there are multiple composites, users can scroll down to see all of them. In more detail, the graphical user interface consists of the following elements:

- (1) *Scene view*
 - Shows the customised or “altered” scene
- (2) *Camera view*
 - Camera presets to quickly change the angle and distance from the rendered objects
- (3) *Prompt*
 - Text field where users can enter a prompt. The send button (i.e. `Online Prompt Submit`) will start the prompt processing and create composites immediately afterwards.
- (4) *Scene and composite controllers*
 - The `Re-execute` button restarts the composite processing. Users can click on it after making changes to selectors and layers to see the new results.
 - The `Clear All` button removes all existing composites and their content.
 - The `+ NEW COMP` button creates new empty composites.
- (5) *Composite interface*
 - Users can change the composite's name (for reference only).
 - Users can remove (delete) a composite.
 - By using the arrows, users can move composites up and down to change the order in which they are executed (top ones first).
 - Users can choose a selector type from a drop-down (currently only conditional selectors)
 - Users can choose which layers to add from a drop-down menu.

3.6 Example Workflows

We demonstrate the system's capabilities through two scenarios that highlight different interaction patterns.

3.6.1 Scenario 1: Simple Visual Filtering. The goal is to highlight tall buildings (>10 stories) in red. Interaction using the LLM-only approach would look as follows:

- (1) User types: “*Colour all buildings over 10 stories red*”
- (2) System generates composite with:
 - Selector: `storeysAboveGround > 10`
 - Layers: `Draw as geometry + Position on map + Set colour (red)`
- (3) Scene updates showing only tall buildings in red
- (4) User adjusts colour shade using the GUI colour picker

This scenario demonstrates how LLM enables rapid task completion while a GUI provides precise control for refinement. The resulting composite is illustrated in Figure 4, showing the resulting view

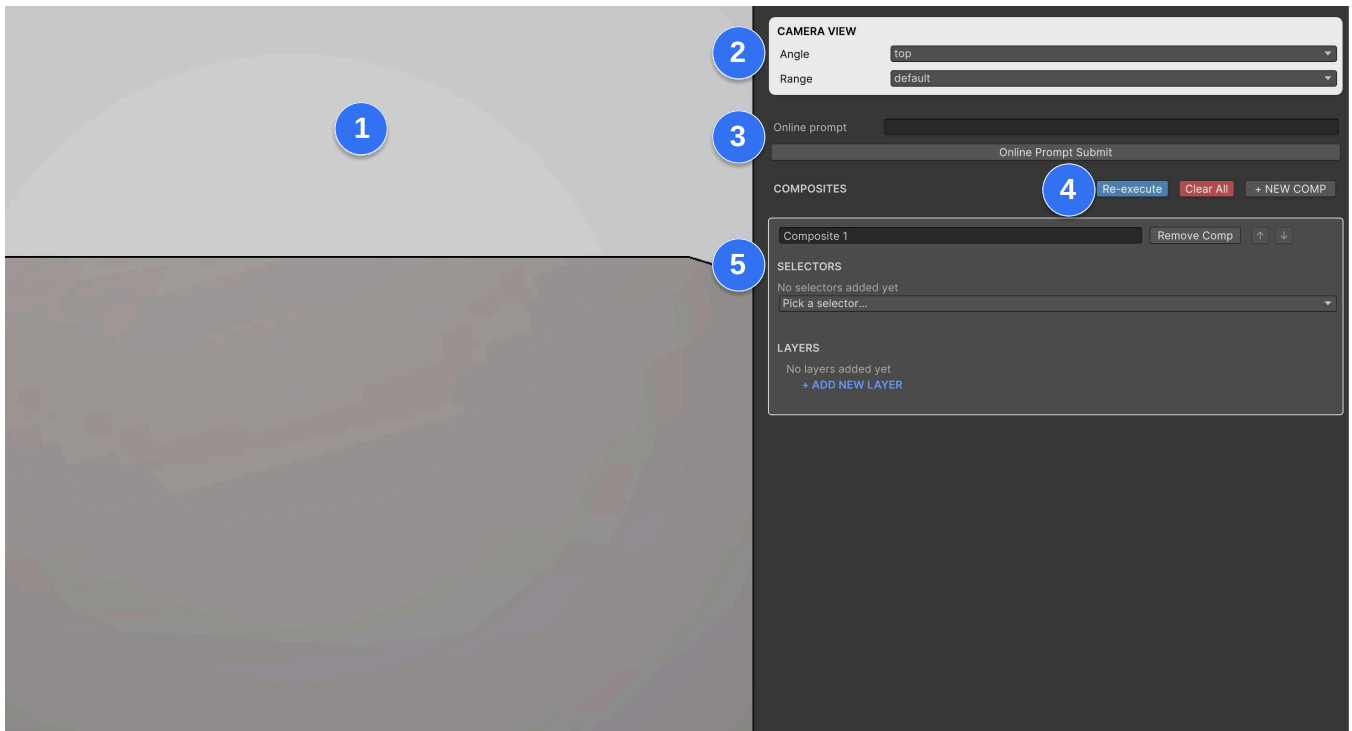


Figure 3: Main components of the graphical user interface

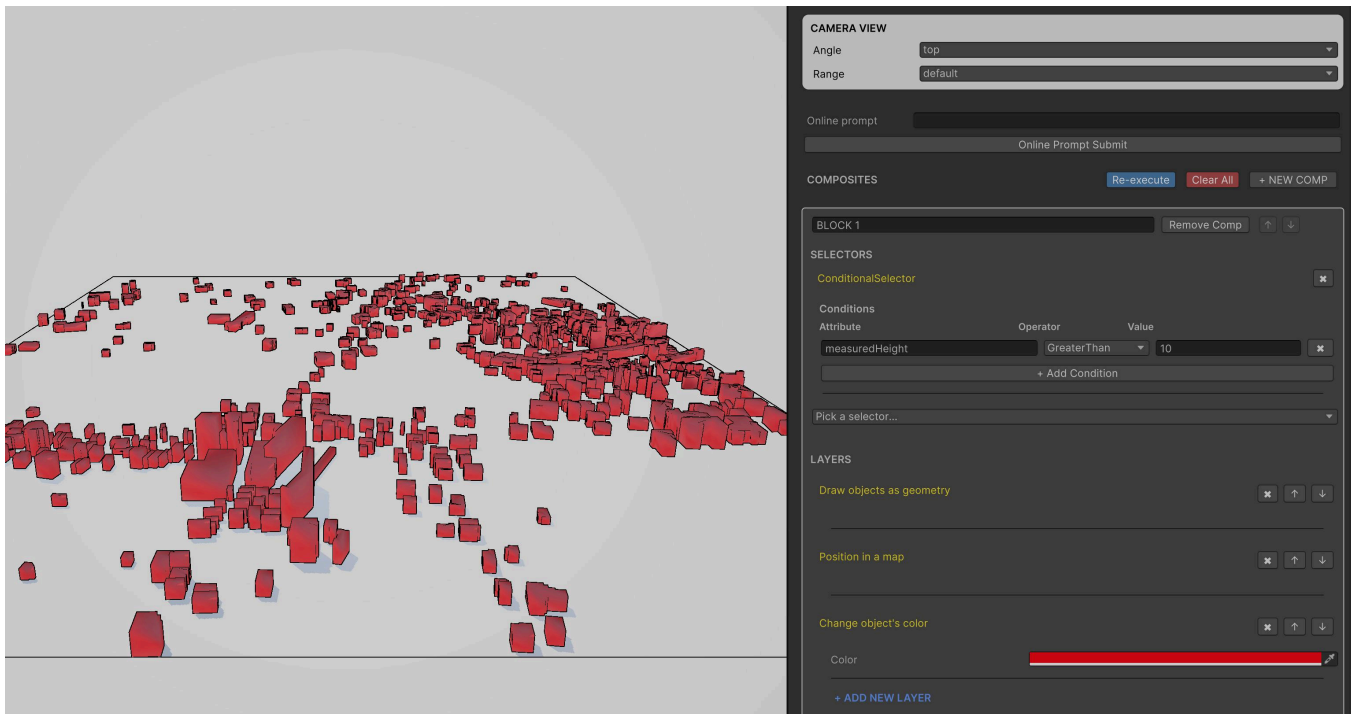


Figure 4: Composite with one selector and three layers; only buildings with more than 10 floors are rendered and coloured

of a city after filtering out smaller buildings and colouring the remaining ones in red.

3.6.2 Scenario 2: Complex Multi-Composite Customisation. The goal is to visualise school-hospital relationships in a city. A user’s interaction following the hybrid approach would look as follows:

- (1) *LLM Setup.* User prompts “Show all buildings in the city”
 - Creates base visualisation with all buildings in grey
- (2) *LLM Refinement.* User prompts “Show schools in blue and connect to closest hospital”
 - System generates a selector for educational facilities
 - Adds layers for colouring and connection lines
- (3) *GUI Addition.* User manually creates new composite:
 - Selector: usage = “medical”
 - Layer: Set colour (red)
- (4) *GUI Fine-tuning.*
 - Doubles line width for better visibility
 - Changes line colour to yellow
 - Enables distance labels (discovered through interface exploration)

This scenario shows how hybrid interaction enables rapid prototyping and detailed customisation, with users discovering features through GUI exploration that enhance their original intent. Our solution eliminates the need for coding and platform-specific expertise while still providing advanced, fully customisable capabilities.

4 User Study

To evaluate how different interface types affect user performance and behaviour when customising 3D environments, we conducted a controlled user study comparing three interaction conditions: GUI-only, LLM-only (prompts) and hybrid. We examined task performance, error patterns and subjective preferences across a diverse set of 3D customisation tasks.

4.1 Study Design

We employed a within-subjects design where each participant completed tasks with all three interfaces. To control for learning and order effects, we designed a counterbalanced study structure:

Task Organisation. We created 12 tasks, divided into three groups (A, B, C), each with 4 tasks. Each group was designed to have similar overall difficulty while varying the complexity of individual tasks. Some tasks required simple single-layer transformations, while others demanded multiple composites with complex selector conditions.

Counterbalancing. Each participant completed three sessions, with each session pairing one task group with one modality as illustrated in Figure 5. The assignment ensured that:

- Every participant experienced all three modalities.
- Every participant completed all three task groups.
- No participant repeated any modality or task group.
- Every participant’s combination of task order and group-modality pairing was unique (e.g. more than one user completed task group A with the GUI-only modality, but only one started with GUI-only, then used LLM-only for group C, and then mixed modality for group B).

Study Design: Within-Subjects Counterbalanced Assignment

Each participant completed 3 sessions (one per modality) with different task groups

Participant	Session 1 (Tasks 1-4)		Session 2 (Tasks 5-8)		Session 3 (Tasks 9-12)	
	Task Group	Modality	Task Group	Modality	Task Group	Modality
P1	A	GUI-only	B	LLM-only	C	Hybrid
P2	B	LLM-only	C	Hybrid	A	GUI-only
P3	C	Hybrid	A	GUI-only	B	LLM-only
P4	B	GUI-only	C	LLM-only	A	Hybrid
P5	C	LLM-only	A	Hybrid	B	GUI-only
P6	A	Hybrid	C	GUI-only	B	LLM-only
P7	C	GUI-only	B	Hybrid	A	LLM-only
P8	A	LLM-only	B	Hybrid	C	GUI-only
P9	B	Hybrid	A	LLM-only	C	GUI-only
P10	A	Hybrid	C	LLM-only	B	GUI-only
P11	B	GUI-only	C	Hybrid	A	LLM-only
P12	C	LLM-only	A	GUI-only	B	Hybrid

Modality Legend:

■ GUI-only
 ■ LLM-only
 ■ Hybrid

Counterbalancing Summary:

- Starting modality: GUI-only (n=4), LLM-only (n=4), Hybrid (n=4)
- Each task group paired with each modality 4 times
- 7 unique modality order sequences
- Tasks within sessions: fixed order (1→2→3→4)

Figure 5: Counterbalance overview

While task groups were randomised across participants, tasks within each group maintained fixed ordering (always 1→2→3→4) to preserve the designed difficulty progression.

4.2 Participants

We recruited 12 participants (aged 22 to 33, $M = 27.5$, $SD = 3.2$) through university and professional networks. Participants varied in their experience with LLM-powered tools, which we quantified on a 4-point Likert scale: 0 (never use genAI tools, n=3), 1 (use 1–2 days per week, n=4), 2 (use 3–4 days per week, n=3), and 3 (use 5–7 days per week, n=2). All participants met our inclusion criteria:

- *Technical Proficiency.* All participants self-identified as “tech-savvy”, comfortable with general computer interfaces and web applications, though none were frequent users of 3D software. This profile matches our target user population: domain experts who need to work with 3D visualisations but lack specific 3D training.
- *Language Proficiency.* All participants were fluent in English (speaking, listening, reading and writing), but none were native English speakers. This was particularly relevant for the LLM condition, in which participants needed to formulate natural language queries and understand system responses.

4.3 Tasks

The tasks were designed to reflect semi-realistic 3D scene customisation scenarios using urban data from the Meguro district in Tokyo. Each task required participants to achieve specific visual outcomes by combining selectors, conditions and layers. Some examples of the tasks that participants had to address are:

- “Make all school buildings blue” (simple: one selector, one colour layer).
- “Place all buildings in a grid and add labels showing building usage” (complex: multiple layers, ordering matters).

- “Colour residential buildings by height and connect schools to nearest hospitals” (complex: multiple composites, spatial relationships).

Tasks varied along several dimensions, including the number of composites required (1–3 composites per task), selector complexity (single versus multiple conditions with AND logic), layer complexity (1 to 4 layers per composite) and order sensitivity (whether layer ordering affected the outcome).

After completing the 12 timed tasks, participants attempted a 13th open-ended, exploratory task that could be solved in multiple ways. This untimed task allowed participants to freely explore the system using their preferred modality, providing insights into natural preference patterns when users are not constrained by experimental conditions.

4.4 Conditions

There were three possible conditions, including the point-and-click-based GUI-only, the prompt-based LLM-only and the hybrid condition in which users could choose to use both paradigms.

GUI-only. Participants exclusively used the graphical interface to create composites, define selectors through dropdown menus and input fields, and add layers from a predefined list. This condition tested the traditional direct manipulation interaction.

LLM-only. Participants expressed their intent entirely through natural language queries typed into a text field. Through Mistral’s API (Mistral Small 3.2), the system interpreted the prompt and generated appropriate composites, selectors, and layers. Participants could see the resulting GUI configuration but could not modify it directly; changes could only be made by submitting new or refined prompts.

Hybrid. Participants could freely combine both modalities. They might start with a natural language query to rapidly create initial configurations, then refine parameters via GUI controls, or vice versa. This condition tested whether users could effectively mix both modalities and how they did it.

4.5 Procedure

Before beginning the time-tracked tasks, participants were given an opportunity to ask questions and explore the interface at their own pace. The procedure unfolded as follows:

- (1) *Introduction (10 minutes).* Participants received an overview of the types of 3D scenes they were going to interact with and the customisation challenge.
- (2) *Demonstration (15 minutes).* The facilitator demonstrated all three interfaces using a simple example task.
- (3) *Reference Materials.* Participants received and reviewed a list of available layers with descriptions and a list of entity attributes with example values. These materials remained available throughout the study.
- (4) *Practice Tasks (10–20 minutes).* Participants completed practice tasks until they successfully created a composite with selector and layers independently.

Afterwards, participants worked through a series of tasks organised into three separate sessions:

- (1) *Session Structure.* Each participant completed three sessions (one per modality), with optional breaks between sessions.
- (2) *Task Execution.* For each task, the participant read the task description, the timer started when they began interaction, participants interacted with the system, and the timer stopped upon task completion or participant surrender.
- (3) *Technical Support.* The facilitator only intervened for technical failures (not user errors), pausing the timer during troubleshooting.

Studies were conducted with one researcher in charge for the whole session. Users were anonymised; the captured data was associated with a code rather than users’ names. No voice or image of the participants was recorded. Time was manually tracked, as well as user “errors”. We recorded on-screen interactions, which were used later to confirm task completion time and the number of errors. The researcher noted behaviours and patterns during the sessions, as well as the feedback users provided while completing the tasks. After the study, participants ranked modalities by preference, perceived efficiency and sense of control³.

4.6 Measures

We collected both objective performance metrics and subjective assessments:

- *Performance Metrics.* Completion Time (time from task start to successful completion) and Task Success (binary measure of whether the task was completed correctly).
- *Error Types (GUI and Hybrid only).* Missing Layer (failed to add a required layer), Wrong Order (placed layers in incorrect sequence), Wrong Attribute (selected incorrect attribute or value), Typo (correct intent but typing error) and Total Errors (sum of all error types per task).
- *LLM-specific Metrics (NLP and Hybrid).* AI Accuracy (whether AI correctly interpreted and executed user prompt) and Error Recognition (whether users noticed mistakes made by AI).
- *Subjective Measures.* Modality Preference (ranking of three modalities), Perceived Efficiency (which modality users believed was fastest), Perceived Control (which modality provided the greatest sense of control) and Prior AI Experience (self-reported frequency of AI tool usage on a 0–3 scale).

4.7 Analysis Approach

We employed non-parametric statistical methods for our analysis due to the study’s small sample size ($N = 12$), repeated-measures design and the presence of outliers in completion time distributions.

For omnibus tests comparing the three modalities, we used the Kruskal-Wallis H test. When significant differences were detected, we conducted post-hoc pairwise comparisons using Mann-Whitney U tests with Bonferroni correction (adjusted $\alpha = 0.0167$) to control for Type I error. We report effect sizes using the rank-biserial correlation (r) for pairwise comparisons and partial eta-squared (η^2) for variance decomposition. To examine learning effects, we analysed completion time across sessions (1–3) using Kruskal-Wallis tests and computed Spearman’s rank correlation coefficient (ρ) between task order (1–12) and completion time.

³Full study dataset and LLM fine-tuning: <https://doi.org/10.5281/zenodo.19476043>

5 Results

5.1 Overall Performance Across Modalities

We found significant differences in completion time across the three modalities (Kruskal-Wallis $H = 15.53$, $p < 0.001$) as illustrated in Figure 6. Pairwise comparisons revealed that LLM-only ($M = 134.5s$, $SD = 72.3$) was significantly faster than GUI-only ($M = 202.2s$, $SD = 99.9$; $U = 638$, $p < 0.001$, $r = 0.45$). Prompts were also faster than mixed ($M = 164.5s$, $SD = 72.8$; $U = 833$, $p = 0.020$, $r = 0.28$), though this difference did not survive Bonferroni correction. The difference between GUI-only and hybrid was not significant ($U = 1412$, $p = 0.057$, $r = 0.23$). The LLM interface demonstrated a 34% speed advantage over GUI-only and an 18% advantage over hybrid. Effect sizes ranged from small to moderate ($r = 0.23$ – 0.45). To contextualise, users could complete approximately 1.5 tasks using prompts in the time required to complete a task using the GUI only.

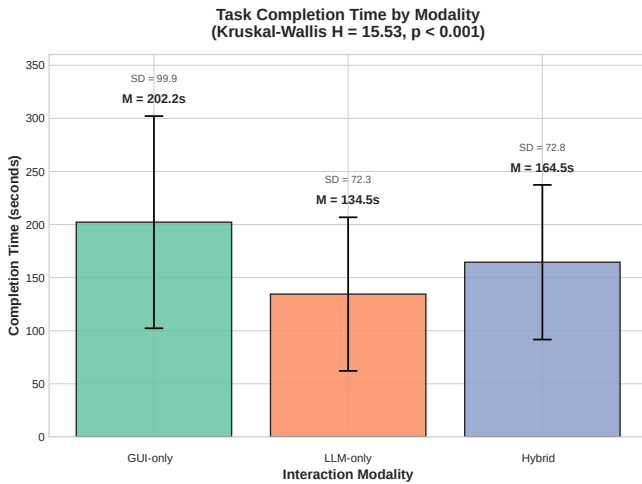


Figure 6: Task completion time by modality

5.2 Learning Effects Across Sessions

As shown in Figure 7, an analysis of the session-by-session performance revealed a significant learning effect ($H = 7.61$, $p = 0.022$), with mean completion times decreasing consistently from Session 1 ($M = 198.8s$) through Session 2 ($M = 156.9s$) to Session 3 ($M = 145.5s$). Spearman correlation confirmed a negative relationship between task order and completion time ($\rho = -0.24$, $p = 0.004$). The examination of learning curves by modality revealed differential improvement rates. The GUI-only condition showed the steepest learning curve ($287s \rightarrow 169s$, 41% improvement from Session 1 to 3), while LLM-only showed the shallowest one ($129s \rightarrow 120s$, 7% improvement). The hybrid interface fell between these extremes ($181s \rightarrow 140s$, 22% improvement).

5.3 Factors Influencing Performance

To understand the relative importance of different factors, we calculated eta-squared (η^2) effect sizes through variance decomposition. Task content explained the most variance ($\eta^2 = 0.124$, 12%), followed by modality ($\eta^2 = 0.103$, 10%) and session ($\eta^2 = 0.070$, 7%).

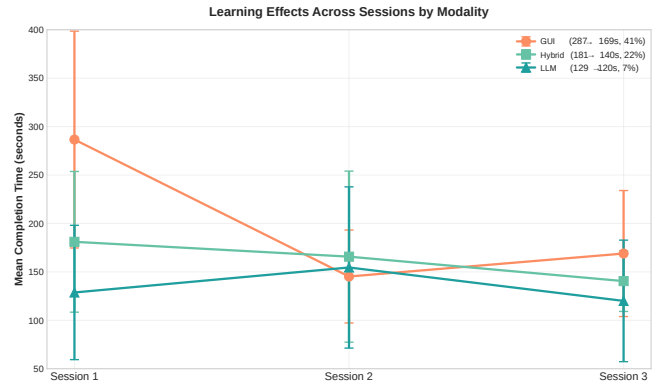


Figure 7: Learning effect

Task groups varied substantially in difficulty, with completion times ranging from 117s (task 8) to 230s (task 11), a difference of a factor of 2. Task 11 was particularly challenging ($M = 230s$, 37% above average), requiring multiple nested conditions and precise layer ordering.

5.4 Error Analysis and LLM Performance

For GUI-only and hybrid interfaces, where errors could be directly attributed to user actions, 57% of task attempts were completed error-free. Errors were distributed across tasks, with task 5 and task 4 showing the highest error rates (15% and 13% of all errors, respectively), but these are not significantly higher than the average. While there is a slight trend for first tasks within groups (1, 5, 9) to be slower ($M = 179.8s$) and with more errors ($M = 0.71$ errors/task) than fourth tasks (4, 8, 12; $M = 147.5s$, $M = 0.50$ errors/task), these differences were not statistically significant (completion time: $H = 3.42$, $p = 0.331$; errors: $H = 1.26$, $p = 0.738$). The variation is more likely due to individual task characteristics rather than position effects.

In the LLM-only condition, the LLM system produced configurations requiring correction in 33 of 48 interactions (69% error rate). Of these LLM errors, participants identified and corrected 27 (82% recognition rate). Note that we did not conduct a formal error taxonomy for the LLM's output. However, we identified the most frequent error type to be missing layers. The LLM would generate composites with correct selectors and some layers, but omit one or more required layers; in most cases, the layer that positioned objects on a map or the one that rendered them in their original form. This was not as common when the prompt required objects to be placed in a grid or rendered as cubes. Normally, participants identified these omissions by inspecting the generated composite in the GUI panel before executing it. Otherwise, they identified them when the composite was not rendered (i.e. something was missing) or the result was different from what was expected. Less frequently, the LLM produced JSON responses that could not be parsed, preventing composites from being created altogether. In these cases, the timer was paused and participants were asked to resubmit their prompt.

5.5 Starting Modality Effects

Error rates did not vary significantly by starting interface type ($H = 1.47$, $p = 0.479$). As highlighted in Figure 8, participants who first experienced the GUI-only interface ($M = 0.62$ errors per task) showed similar error rates to those who started with LLM-only ($M = 0.62$) or hybrid ($M = 0.44$).

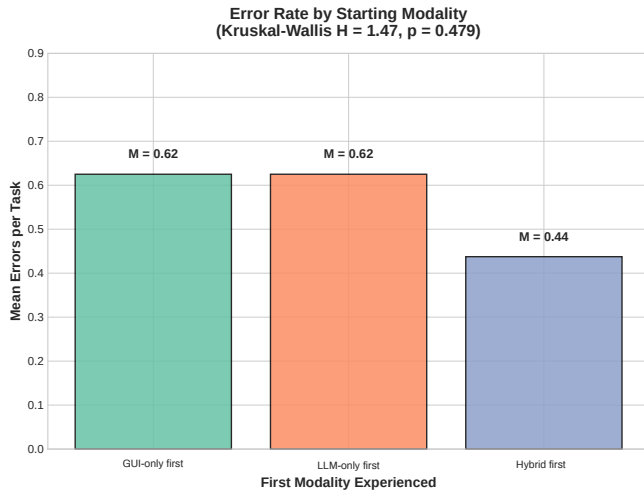


Figure 8: Error rate by modality

However, we observed a trend in LLM error recognition rates. Users who first experienced GUI-based interaction (either GUI-only or hybrid) showed higher rates of recognising LLM errors (93%) than those who started with prompts (54%). This difference was marginally significant ($U = 26.50$, $p = 0.058$) and warrants further investigation to determine whether prior GUI exposure helps users build mental models that support LLM error detection.

5.6 User Perceptions

Post-study structured interviews revealed clear patterns in user preferences and perceptions:

- *Modality Preference.* Hybrid was preferred by 8 of 12 participants (67%), GUI-only by 3 (25%), and LLM-only by 1 (8%). 10 of 12 participants (83%) ranked prompts as their least preferred modality.
- *Perceived Control.* 10 of 12 participants (83%) identified GUI-only as providing the greatest sense of control, with 2 (17%) choosing the hybrid option. No participants selected the LLM-only modality.
- *Perceived Efficiency.* Opinions were diverse: 5 participants (41.5%) perceived hybrid as most efficient, 5 (41.5%) chose LLM-only and 2 (17%) chose GUI-only.
- *Perception-Performance Gap.* Only 4 of 12 participants (33%) correctly identified the objectively fastest modality. We found no significant relationship between stated preference and actual performance ($U = 109$, $p = 0.66$).

5.7 Prior LLM Experience

Prior LLM experience, which was captured at the end of each study, did not predict the performance of the LLM interface. Spearman

correlations between self-reported LLM experience (0–3 scale) and completion time were non-significant for both hybrid ($\rho = -0.14$, $p = 0.34$) and LLM-only ($\rho = 0.01$, $p = 0.94$) conditions. Users with no prior LLM experience ($M = 142s$) performed comparably to daily LLM users ($M = 145s$).

5.8 Exploratory Task and Behavioural Observations

In the untimed exploratory task (task 13), where participants could freely choose their interaction approach, 10 of 12 participants (83%) chose the hybrid approach and 2 (17%) chose to use GUI-only. No participants chose to work exclusively with prompts.

Among participants using hybrid interaction, a consistent workflow pattern emerged: participants initiated tasks with a natural language prompt, then switched to GUI controls for refinement and adjustment. During interviews, participants explained that prompts served two purposes: rapid initialisation and the discovery of system capabilities. Several noted using prompts to “see if the system had good suggestions” when they were uncertain how to approach a task.

We observed participants adapting their prompt language over the course of the study. Initial prompts used natural, conversational phrasing (e.g. “show me buildings with a basement”), whereas later prompts increasingly adopted technical vocabulary aligned with system attributes (e.g. “show me buildings with more than one story below ground”). Similarly, participants shifted from semantic descriptions (“colour houses”) to explicit attribute references (“colour objects whose ‘usage’ equals ‘housing’”).

6 Discussion

Our findings reveal three fundamental tensions in hybrid GUI-LLM interfaces for 3D visualisation: *efficiency versus learning*, *automation versus mental model formation*, and *speed versus perceived control*. We discuss each tension, connecting our observations to the broader HCI literature, before presenting some design recommendations based on these findings.

6.1 AI-reduced Learning Effect: Efficiency Versus Learning

The differential learning curves across modalities, 41% improvement for GUI-only versus only 7% for LLM-only, present a counterintuitive finding: the most efficient interface may be the worst for skill development.

This pattern aligns with Bjork and Bjork’s concept of “desirable difficulties” in learning [3]. Their research demonstrates that conditions which slow initial performance often enhance long-term retention and transfer. The GUI-only interface, by requiring explicit specification of each selector and layer, appears to create precisely such productive friction. Users must engage with the system’s conceptual structure, understanding what selectors do, how layers compose and why ordering matters, rather than delegating this reasoning to the AI.

The paradox challenges assumptions prevalent in natural language interface research. Surveys of V-NLI systems emphasise how natural language “enables users to focus on their tasks rather than

having to” learn complex tools [21]. Our findings suggest this framing may be incomplete: focusing users on tasks by abstracting away interface mechanics may simultaneously deprive them of learning opportunities. The efficiency gains are real but may come at a hidden cost.

This tension is particularly relevant given Jankowski and Hachet’s observation that successful game interfaces employ a “*layered or spiral approach to learning*” [12], progressively revealing complexity as users develop expertise. Our data suggests that AI-powered interfaces may inadvertently flatten this spiral, offering immediate access to advanced capabilities without the foundational understanding that makes those capabilities meaningful.

6.2 Scaffolding and Mental Model Formation

While we did not find a significant scaffolding effect on error rates across modalities as discussed in Section 5.5, we observed a suggestive trend in AI error recognition that connects to Bruner’s theory of cognitive scaffolding [5]. Bruner argued that learning should progress from concrete, enactive representations to more abstract, symbolic ones. In our context, the GUI represents concrete interaction (visible parameters, explicit choices and immediate feedback), while prompts represent symbolic abstraction (intentions expressed in language, translated by AI into system operations).

Users who first experienced GUI-based interaction (GUI-only or hybrid) showed higher rates of recognising LLM errors (93%) compared to those who started with LLM-only (54%), though this difference was only marginally significant ($p = 0.058$). This trend suggests that concrete GUI interaction may help users develop mental models that support evaluation of LLM-generated output, even if it does not directly reduce their own error rates. However, this finding requires further investigation with larger sample sizes to be confirmed.

This pattern connects to Wang et al.’s work on LLM-powered conversational interfaces, which notes that such approaches are “*beneficial for accessibility*” but acknowledges that users still need to understand the underlying system to use it effectively [24]. Our exploratory findings suggest that GUI exposure may support this understanding, particularly for evaluating LLM output.

The high AI error rate we observed (69%) underscores the importance of this evaluative capacity. Users needed mental models not just to operate the system, but to recognise when AI-generated configurations deviated from their intent. The overall 82% error recognition rate indicates that users can catch AI mistakes, and our trend data suggests that prior GUI exposure may enhance this capability, but this hypothesis requires further analysis.

6.3 The Control-Efficiency Tension

Our perception data revealed an interesting disconnect: using LLM-only was objectively faster, yet 83% of participants ranked it as the least preferred interface. Meanwhile, 83% identified GUI-only as providing the greatest sense of control, and 67% preferred the hybrid approach overall. Users consistently chose perceived control over measured efficiency.

This pattern challenges purely performance-based evaluation of interfaces. If users cannot accurately assess which tools make them fastest (only 33% correct), and they systematically prefer slower

tools that offer greater agency, what should designers optimise for? Our findings suggest that subjective control and objective efficiency represent distinct dimensions that may trade off against each other.

The preference for hybrid interaction, chosen by 67% in stated preferences and 83% in unconstrained behaviour, suggests users seek to balance these dimensions rather than maximise either. Min et al.’s work on malleable interfaces similarly found that users value the ability to customise their interaction approach, not just the outcomes they achieve [14]. The hybrid modality may succeed precisely because it preserves user agency: users can choose when to delegate to AI and when to take direct control.

It is important to consider that the high percentage of prompts requiring correction (69%) may have influenced users’ opinions and preferences regarding the LLM modality. More research is needed to see how this preference would be impacted by more accurate LLM fine-tuning. Still, in the unconstrained exploratory task, 83% of participants chose to incorporate prompts into their workflow despite having experienced LLM errors throughout the study. This suggests that users found value in natural language interaction even when it was not completely reliable, given that they had GUI controls available for verification and correction.

This has implications for how we interpret the efficiency advantages of natural language interfaces documented in prior work. Studies showing that NLP enables faster task completion [21, 25] capture an important dimension of interface quality, but our findings suggest this dimension does not fully determine user preference or satisfaction. Users working with complex 3D visualisations appear to value understanding and control alongside, or even above, raw speed.

6.4 Prompts as Discovery and Initialisation Tools

The emergent *prompt-then-refine* workflow observed in our exploratory task offers insight into how users naturally integrate multiple modalities. Rather than using prompts as a replacement for GUI interaction, participants employed them strategically: for rapid initialisation at task start and for discovery when uncertain about system capabilities.

The discovery function is particularly noteworthy. Participants reported using prompts to “*see if the system had good suggestions*” when they lacked clear ideas about how to approach a task. This aligns with findings from LowCoder, which found that “*language models helped users discover previously-unknown operators in 75% of tasks*” [19]. Prompts serve not only as an efficient input modality but also as a means to query the system’s knowledge of what is possible.

The language adaptation we observed, users shifting from natural descriptions (“*colour houses*”) to technical vocabulary (“*colour objects whose ‘usage’ equals ‘housing’*”), suggests a learning process distinct from the efficiency-focused framing common in NLP interface research. Users were not simply finding faster ways to express fixed intentions; they were developing more precise mental models of the system’s attribute structure. This vocabulary acquisition may represent a form of learning that prompts *do* support, even as they reduce learning about interface mechanics.

6.5 Design Recommendations

Based on our findings, we propose *seven design recommendations (DR1–DR7)* for hybrid 3D visualisation interfaces. Each implication is grounded in specific empirical observations and connected to relevant prior work.

DR1: Consider GUI-first Onboarding. Our exploratory finding that GUI-first users showed higher LLM error recognition rates (93% versus 54%, $p = 0.058$) indicates that onboarding can benefit from beginning with explicit GUI interaction before introducing LLM assistance (see Section 5.5). While this trend might still require further confirmation, it aligns with Bruner’s scaffolding principle [5] and the progressive disclosure patterns identified by Jankowski and Hachet in successful game interfaces [12]. Therefore, systems might consider *GUI-first* onboarding sequences to help users build mental models that support effective evaluation of LLM-generated output.

DR2: Design for High AI Error Rate. With 69% of prompts requiring correction (see Section 5.4), hybrid interfaces must assume LLM might frequently fail. Interfaces should prioritise efficient error-correction workflows: clear views showing what the LLM changed, easy rollback mechanisms and transitions from LLM output to manual refinement. The DataFormulator approach, which combines “unbounded expressiveness” with GUI “precision and affordance” [25], offers one model for this integration.

DR3: Leverage User Error Detection Capability. The 82% error recognition rate (see Section 6) demonstrates that users are capable reviewers of LLM output, provided they have adequate mental models (see DR1). Interfaces should support this human oversight role rather than attempting to hide LLM uncertainty. Design patterns might include confidence indicators, alternative suggestions, or explicit “review and confirm” steps before LLM changes take effect. This recommendation aligns with broader calls for human-centred AI that maintains meaningful human control [1].

DR4: Support the Prompt-Then-Refine Workflow. The consistent pattern of prompt initialisation followed by GUI refinement (see Section 5.8) represents an emergent best practice that deserves explicit design support. Interfaces should make transitions between modalities frictionless: prompts should generate editable GUI representations, and GUI states should be describable in natural language for further prompting. The bidirectional consistency in our selector-layer architecture, where any state can be reached through either modality, provides one implementation approach.

DR5: Enable Prompts as Discovery Tools. Users employed prompts not just for efficiency but to explore system capabilities (see Section 5.8). Interfaces should explicitly support this discovery function, perhaps through “suggestion mode” features that show what the system *could* do rather than immediately executing changes. This recommendation extends findings from LowCoder on LLMs helping users discover unknown operators [19] to the 3D visualisation domain.

DR6: Support Vocabulary Learning. The observed shift from natural to technical language (see Section 5.8) suggests users benefit from learning system-specific vocabulary. Interfaces might support this through progressive disclosure of technical terms, autocomplete

suggestions that teach attribute names, or explanations that map natural descriptions to formal selectors. This represents a form of learning that prompts can support, complementing the structural learning provided by GUIs.

DR7: Design for Control, Not Just Speed. The strong preference for perceived control (83% chose GUI-only as the most controllable option) despite the LLM’s greater efficiency (see Section 5.6) suggests that control is a key design objective. Systems should preserve user agency even when offering LLM assistance: making LLM suggestions optional rather than automatic and providing clear mechanisms to override LLM output. This aligns with Miraz et al.’s emphasis on understanding how users “deal with information presented to them” [15] and with Min et al.’s findings about user demand for interface customisation [14].

6.6 Framework Contributions

The selector-layer architecture underlying LCS represents a conceptual contribution that both enabled our empirical findings and extends beyond our specific implementation.

Cognitive Alignment. The observed prompt adaptation patterns, with users shifting toward attribute-based language, suggest the selector-layer model aligns with how users naturally conceptualise 3D customisation tasks. Users think in terms of *what objects* (selectors) and *what changes* (layers), a separation that maps to the architecture’s core abstractions.

Interface Symmetry. Because selectors and layers are declarative rather than procedural, they can be specified through GUI controls, natural language or code, while producing identical results. This symmetry was essential to the hybrid workflow we observed: users could prompt to generate a configuration, then refine it in the GUI without losing information or capabilities. The pattern shares conceptual similarities with declarative visualisation grammars such as Vega-Lite [20].

Composability and its Costs. The layered execution model enables iterative refinement, allowing users to add, remove or reorder transformations incrementally. However, our error analysis revealed that layer ordering was a significant source of difficulty: task 4, task 7 and task 11, which required precise sequencing, accounted for 30% of all errors. This suggests that while composability supports flexibility, it also introduces cognitive complexity that neither GUI nor NLP fully mitigate.

Domain Generalisation. While we implemented LCS for urban digital twins, the selector-layer pattern is domain-agnostic. It might be applied to scientific visualisation (selecting molecules by properties, applying rendering styles), manufacturing (selecting components by specifications, applying inspection views) or data analytics (selecting records, applying visual encodings).

The system could also operate at different scales. At a finer granularity, the same pattern could apply within a single building, where users customise the visual properties of rooms, furniture or pipelines based on attributes such as function, material or maintenance status. At the object level, a user inspecting a vehicle model could select components based not only on basic properties like type or position, but also on functional attributes such as thermal

exposure or physical contact relationships, and apply visual layers to highlight patterns across those dimensions. In each case, the core interaction model remains the same: users define *what* to affect through semantic selectors and *how* to affect it. What would change across domains is the set of available attributes, as well as the types of selectors and layers, depending on what is relevant in each case.

The presented architecture can also be extended to temporal data. With live data feeds, selectors would operate on real-time attribute values, for instance, selecting all buildings whose current energy consumption exceeds a threshold, with the visual result updating as values change. The selector-layer separation remains applicable: the temporal dimension affects *which* objects are selected or *how* layer values are computed, but does not require changes to the interaction model.

Note that Barricelli and Fogli's review of digital twins in HCI [1] identified visualisation accessibility as a critical gap, and the presented selector-layer pattern offers one architectural approach to addressing this gap across domains.

7 Limitations

Our study has several limitations that should be considered when interpreting the findings.

Sample Size and Generalisability. With 12 participants, our sample provides initial evidence but limited statistical power and generalisability. All participants were tech-savvy non-experts with similar educational backgrounds, recruited primarily through university networks. While this profile matches our target user population, it does not represent the full diversity of potential 3D software users, including older professionals, those with lower technical confidence or domain experts in fields like urban planning or facilities management.

The tech-savvy profile of our participants likely facilitated faster adaptation to the GUI and a greater willingness to inspect LLM-generated composites for errors, even when participants' prior LLM experience varied. With a less technically confident population, both the GUI learning curve and the difficulty of evaluating LLM output could increase.

Language Proficiency. All participants were fluent but non-native English speakers. This may have affected completion times in the prompts and mixed conditions, where formulating effective natural language queries requires additional cognitive effort. This might have disadvantaged the LLM-only condition relative to the GUI-only setting. Studies with native English speakers or with LLM support for other languages, might reveal slightly different preference patterns.

Study Duration and Learning Effects. Our cross-sectional design captured performance within a single session, but we cannot speak to how modality preferences and effectiveness might evolve with extended use. The observed trend in AI error recognition and the reduced learning effect when using AI may evolve differently over weeks or months of regular system use.

Task Design. While our tasks reflected realistic digital twin customisation scenarios, they remained structured experimental tasks

with clear success criteria. Real-world use often involves more exploratory, open-ended goals, where users may not have a precise outcome in mind.

LLM Implementation. Our natural language processing relied on Mistral Small 3.2, which was trained on approximately 150 few-shot examples. More extensive fine-tuning, larger training sets or more capable models could improve interpretation accuracy and reduce users' error rates, which in turn might affect users' preferences.

System Specificity. Our findings are based on the evaluation of a single system (LCS) in its current implementation. While we believe the patterns we observed offer insights relevant to hybrid interface design more broadly, different architectural choices, domain contexts or interaction designs may yield different results.

8 Conclusion and Future Work

In this study, we explored how the selector-layer architecture can be used in 3D scenes with fixed values. Nevertheless, the architecture can be further improved to support dynamic behaviours and temporal patterns in 3D visualisations, thereby enabling its use in simulation systems or digital twins with real-time updates. We will also explore other types of interfaces (e.g. node-based) built on the same underlying framework to improve its flexibility and enable other teams to create new workflows for 3D scene customisation. This includes collaborative features, which may come with their own challenges in keeping all users' actions visible, even when each user is working with a different interface type.

More longitudinal studies are needed to investigate how the observed trend in LLM error recognition and interface types evolves over extended use and whether GUI-first exposure produces lasting benefits for evaluating LLM output in hybrid interfaces.

Finally, we plan to experiment with different domains, including manufacturing and healthcare. In this scenario, users may not highlight entire objects but specific parts of them; for instance, visual treatments may be used to highlight pieces of a car based on their functionality. This low-level analysis of objects will most likely require new approaches to how users select objects beyond spatial area and semantics.

To conclude, we investigated how hybrid GUI-LLM interfaces affect user performance, behaviour and preferences when customising 3D digital twin environments. Through the design and evaluation of the Layered Customisation System (LCS), we provide preliminary empirical evidence that challenges simplistic assumptions about the superiority of AI-assisted interaction. Our controlled study with 12 participants is still limited, but it suggests a fundamental tension between efficiency and learning. While LLM prompts offered a 34% advantage in task completion time (speed) over GUI-only interactions, this efficiency came at a cost: the faster modality appeared to provide fewer opportunities for skill development. Users starting with LLM-only improved only 7% across sessions, compared to 41% improvement with GUI-only.

We observed a trend suggesting that the initial exposure to a modality might influence users' ability to evaluate AI output. Participants who first experienced GUI-based interaction showed higher rates of recognising AI errors, though this finding requires confirmation in future studies. The observed trend suggests that explicit

interface interaction may help build mental models that support effective oversight of AI-generated configurations. We also documented a perception-performance gap: only 33% of the participants correctly identified their objectively fastest modality. Users consistently preferred modalities that offered perceived control over those that maximised speed, with 83% identifying GUI-only as most controllable, while prompts were least preferred despite being fastest.

Our observations of user behaviour in the hybrid condition revealed interesting modality-switching strategies. Users employed GUI-only for known tasks but turned to prompts for discovery, using the LLM to uncover layers they did not know existed. These behaviours informed our seven design recommendations, which emphasise considering GUI-first onboarding, designing for high AI error rates, and supporting the prompt-then-refine workflow.

The selector-layer architecture underlying LCS offers a conceptual contribution beyond our specific implementation. By separating *what* to customise from *how* to customise it, the presented declarative model enables modality symmetry, which makes fluid hybrid interaction possible.

Our work suggests that hybrid GUI-LLM interfaces for 3D visualisation are not simply about offering a choice between modalities. The value lies in how these modalities interact: GUIs helping users build mental models that make AI assistance more effective, prompts revealing functionality that users then refine through direct manipulation, and hybrid GUI-LLM interfaces providing both the efficiency of automation and the agency of manual control.

References

- [1] Barbara Rita Barricelli and Daniela Fogli. 2024. Digital Twins in Human-Computer Interaction: A Systematic Review. *International Journal of Human-Computer Interaction* 40, 2 (2024). doi:10.1080/10447318.2022.2118189
- [2] Lonni Besançon, Paul Issartel, Mehdi Ammi, and Tobias Isenberg. 2021. The State of the Art of Spatial Interfaces for 3D Visualization. *Computer Graphics Forum* 40, 1 (2021). doi:10.1111/cgf.14189
- [3] Elizabeth Bjork and Robert Bjork. 2011. Making Things Hard on Yourself, but in a Good Way: Creating Desirable Difficulties to Enhance Learning. In *Psychology and the Real World: Essays Illustrating Fundamental Contributions to Society*. Worth Publishers.
- [4] Richard A. Bolt. 1980. "Put-That-There": Voice and Gesture at the Graphics Interface. *ACM SIGGRAPH Computer Graphics* 14, 3 (1980). doi:10.1145/965105.807503
- [5] Jerome S. Bruner. 1966. Toward a Theory of Instruction. *The Bulletin of the National Association of Secondary School Principals* 50, 309 (1966). doi:10.1177/019263656605030929
- [6] Yining Cao, Peiling Jiang, and Haijun Xia. 2025. Generative and Malleable User Interfaces with Generative and Evolving Task-driven Data Model. In *Proceedings of CHI 2025, SIGCHI Conference on Human Factors in Computing Systems*. Yokohama, Japan. doi:10.1145/3706598.3713285
- [7] Min Chul Cha. 2025. Switching Between Touch and Voice: Factors Influencing Modality Selection in Multimodal Systems. *Ergonomics* (2025). doi:10.1080/00140139.2025.2499200
- [8] Joao L. D. Comba, Nicolau O. Santos, Jonathan C. Rivera, Regis K. Romeu, and Mara Abel. 2023. Data Visualization for Digital Twins. *Computing in Science & Engineering* 25, 2 (2023). doi:10.1109/MCSE.2023.3295968
- [9] T.R.G. Green and M. Petre. 1996. Usability Analysis of Visual Programming Environments: A 'Cognitive Dimensions' Framework. *Journal of Visual Languages & Computing* 7, 2 (1996). doi:10.1006/jvlc.1996.0009
- [10] Xiyun Hu, Dizhi Ma, Fengming He, Zhengzhe Zhu, Shao-Kang Hsia, Chenfei Zhu, Ziyi Liu, and Karthik Ramani. 2025. GesPrompt: Leveraging Co-Speech Gestures to Augment LLM-based Interaction in Virtual Reality. In *Proceedings of DIS 2025, International Conference on Designing Interactive Systems*. Funchal, Portugal. doi:10.1145/3715336.3735769
- [11] Kazuma Inokuchi, Jin Nakazato, Manabu Tsukada, and Hiroshi Esaki. 2023. Semantic Digital Twin for Interoperability and Comprehensive Management of Data Assets. In *Proceedings of MetaCom 2023, International Conference on Metaverse Computing, Networking and Applications*. Kyoto, Japan. doi:10.1109/MetaCom57706.2023.00049
- [12] Jacek Jankowski and Martin Hachet. 2015. Advances in Interaction with 3D Environments. *Computer Graphics Forum* 34, 1 (2015). doi:10.1111/cgf.12466
- [13] Tae Soo Kim, DaEun Choi, Yoonseo Choi, and Juho Kim. 2022. Stylette: Styling the Web with Natural Language. In *Proceedings of CHI 2022, SIGCHI Conference on Human Factors in Computing Systems*. New Orleans, USA. doi:10.1145/3491102.3501931
- [14] Bryan Min, Allen Chen, Yining Cao, and Haijun Xia. 2025. Malleable Overview-Detail Interfaces. In *Proceedings of CHI 2025, SIGCHI Conference on Human Factors in Computing Systems*. Yokohama, Japan. doi:10.1145/3706598.3714164
- [15] Mahdi H. Miraz, Maaruf Ali, and Peter S. Excell. 2021. Adaptive User Interfaces and Universal Usability Through Plasticity of User Interface Design. *Computer Science Review* 40 (2021). doi:10.1016/j.cosrev.2021.100363
- [16] Arpit Narechania, Arjun Srinivasan, and John Stasko. 2021. NL4DV: A Toolkit for Generating Analytic Specifications for Data Visualization from Natural Language Queries. *IEEE Transactions on Visualization and Computer Graphics* 27, 2 (2021). doi:10.1109/TVCG.2020.3030378
- [17] Sharon Oviatt. 1999. Ten Myths of Multimodal Interaction. *Commun. ACM* 42, 11 (1999). doi:10.1145/319382.319398
- [18] Elena Planas, Gwendal Daniel, Marco Brambilla, and Jordi Cabot. 2021. Towards a Model-driven Approach for Multiexperience AI-based User Interfaces. *Software and Systems Modeling* 20, 4 (2021). doi:10.1007/s10270-021-00904-y
- [19] Nikitha Rao, Jason Tsay, Kiran Kate, Vincent J. Hellendoorn, and Martin Hirzel. 2024. AI for Low-Code for AI. In *Proceedings of IUI 2024, International Conference on Intelligent User Interfaces*. Greenville, USA. doi:10.1145/3640543.3645203
- [20] Arvind Satyanarayan, Dominik Moritz, Kanit Wongsuphasawat, and Jeffrey Heer. 2017. Vega-Lite: A Grammar of Interactive Graphics. *IEEE Transactions on Visualization and Computer Graphics* 23, 1 (2017). doi:10.1109/TVCG.2016.2599030
- [21] Leixian Shen, Enya Shen, Yuyu Luo, Xiaocong Yang, Xuming Hu, Xiongshuai Zhang, Zhiwei Tai, and Jianmin Wang. 2023. Towards Natural Language Interfaces for Data Visualization: A Survey. *IEEE Transactions on Visualization and Computer Graphics* 29, 6 (2023). doi:10.1109/TVCG.2022.3148007
- [22] Ben Shneiderman. 1983. Direct Manipulation: A Step Beyond Programming Languages. *Computer* 16, 8 (1983). doi:10.1109/MC.1983.1654471
- [23] Yuan Tian, Weiwei Cui, Dazhen Deng, Xinhuan Yi, Yurun Yang, Haidong Zhang, and Yingcai Wu. 2025. ChartGPT: Leveraging LLMs to Generate Charts From Abstract Natural Language. *IEEE Transactions on Visualization and Computer Graphics* 31, 3 (2025). doi:10.1109/TVCG.2024.3368621
- [24] Bryan Wang, Gang Li, and Yang Li. 2023. Enabling Conversational Interaction with Mobile UI using Large Language Models. In *Proceedings of CHI 2023, SIGCHI Conference on Human Factors in Computing Systems*. Hamburg, Germany. doi:10.1145/3544548.3580895
- [25] Chenglong Wang, Bongshin Lee, Steven M. Drucker, Dan Marshall, and Jianfeng Gao. 2025. Data Formulator 2: Iterative Creation of Data Visualizations, with AI Transforming Data Along the Way. In *Proceedings of the CHI 2025, SIGCHI Conference on Human Factors in Computing Systems*. Yokohama, Japan. doi:10.1145/3706598.3713296
- [26] Jackie (Junrui) Yang, Yingtian Shi, Yuhan Zhang, Karina Li, Daniel Wan Rosli, Anisha Jain, Shuning Zhang, Tianshi Li, James A. Landay, and Monica S. Lam. 2024. ReactGenie: A Development Framework for Complex Multimodal Interactions Using Large Language Models. In *Proceedings of CHI 2024, SIGCHI Conference on Human Factors in Computing Systems*. Honolulu, USA. doi:10.1145/3613904.3642517
- [27] Bowen Yu and Cláudio T. Silva. 2020. FlowSense: A Natural Language Interface for Visual Data Exploration within a Dataflow System. *IEEE Transactions on Visualization and Computer Graphics* 26, 1 (2020). doi:10.1109/TVCG.2019.2934668