Sentient Sandbox: Modifying Worlds with Language Models

Experience immersive VR creation with spoken language, the most natural form of communication.

JOSHUA JUNG, MICHAEL LI, and ERIC BAE

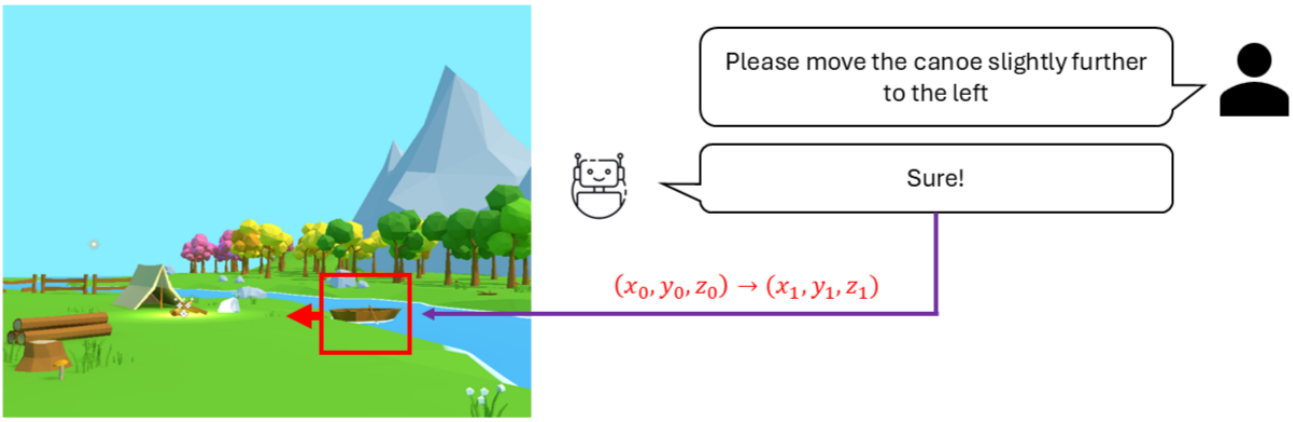

Fig. 1. A diagram of the Sentient Sandbox interface. The user can specify a modification via natural language, and a language agent will translate the command into instructions for modifying the scene. The change in the scene will be reflected in real time in the 3D scene.

Existing methods for 3D scene manipulation in virtual reality often rely on complex and technical interfaces, posing accessibility barriers. The rise of Large Language Models (LLMs) has opened possibilities for using natural language as an interface for technical instructions, which would be more natural and accessible. However, while Large Language Models (LLMs) have shown promise in 2D image editing and static 3D generation, extending this functionality to dynamic, interactive 3D environments remains challenging. Prior work has explored text-based scene generation and object manipulation, but existing approaches often lack fine-grained control and practical features like collision detection and complex object transformations. Furthermore, the inherent ambiguity of natural language makes direct translation into precise 3D instructions difficult. To address these limitations, we propose Sentient Sandbox, a system that enables real-time 3D scene modification through intuitive natural language commands. Sentient Sandbox uses a structured pipeline to translate spoken instructions into tangible 3D environment modifications. We address the challenge of linguistic ambiguity through robust prompt engineering, as well as pre-processing and post-processing steps which serve to ensure consistency. We evaluate the performance of Sentient Sandbox on three separate scenes, and justify the significance of each step in our processing pipeline.

1 INTRODUCTION

As virtual reality (VR) technologies become increasingly mainstream, there is rising demand for more intuitive and accessible methods of content creation and interaction. Traditional interfaces for 3D scene modification often demand technical expertise, creating significant accessibility barriers for many users. The advent of Large Language Models (LLMs) offers a paradigm shift, presenting natural language as an intuitive bridge between human intent and tangible environment modifications across a variety of domains [Ahn et al. 2022; Bubeck et al. 2023; Wan et al. 2024]. This technology holds the potential to democratize 3D design, enabling users of all skill levels to build and iterate on environments with unprecedented ease. From architectural planning to game development, the potential applications are vast and cut across numerous billion-dollar industries. As we move towards more sophisticated virtual environments, the development of robust and intuitive 3D scene manipulation tools becomes indispensable.

While LLMs have demonstrated success in 2D image editing and static 3D generation [Shriram et al. 2025; Zhang et al. 2023], the dynamic and interactive nature of real-time VR scenes poses a unique set of challenges. Existing methods for 3D scene manipulation often fall short in providing the necessary fine-grained control and practical features, such as collision detection, essential for immersive and realistic experiences. Furthermore, overcoming the inherent ambiguity of natural language to achieve precise 3D instruction translation is a critical step towards unlocking the full potential of VR for a wider audience.

Sentient Sandbox is a system which allows modification of 3D virtual worlds in real time via the use of natural language commands. The user will first use spoken language to specify modifications to a 3D virtual world. These prompts are then translated into scene update instructions, through the use of language agents. Our key insight is to build a graph representing objects in the scene and their relationships with each other, which provides useful information that allows LLMs to more accurately interpret user intent.

We evaluate Sentient Sandbox on three separate 3D scenes, and believe it shows promise for future avenues of work on generalized and precise manipulation of 3D scenes.

Authors' address: Joshua Jung, jjung04@cs.washington.edu; Michael Li, ml10872@cs.washington.edu; Eric Bae, ebae3@cs.washington.edu.

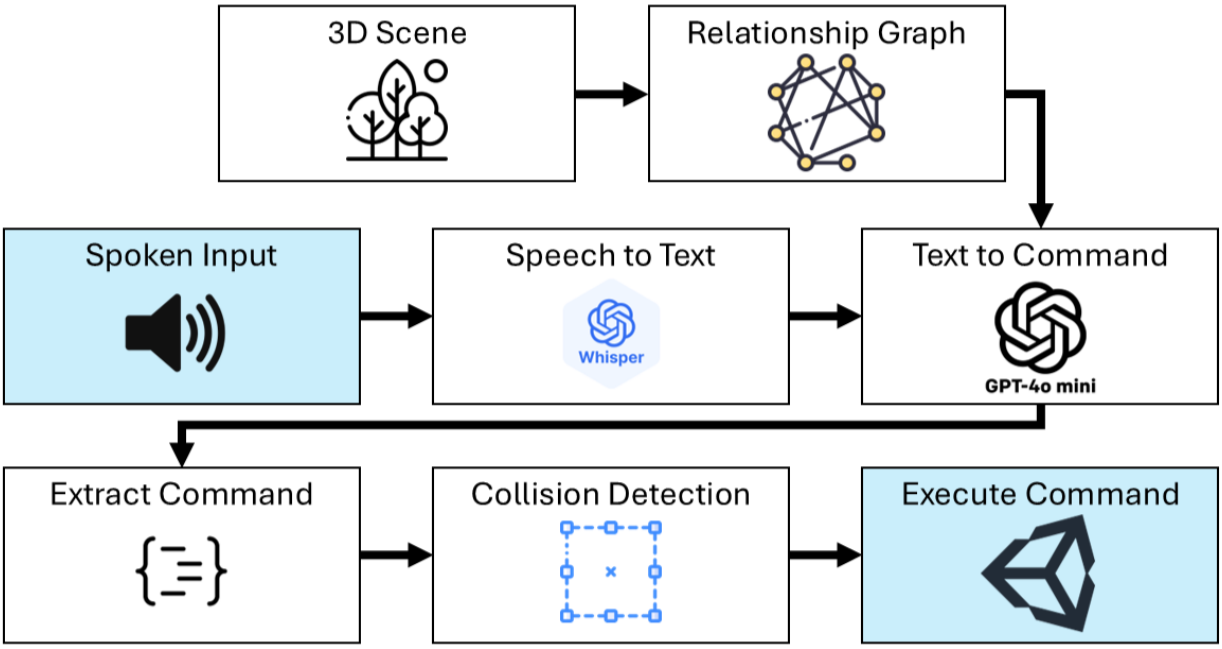

Fig. 2. User interaction in Sentient Sandbox begins with spoken input, which is converted to text via Whisper. The natural language command is then transformed into technical commands using GPT-4o mini, with object relationships informed by a pre-computed graph. This command is checked for possible collisions, ensuring realistic and coherent modifications. Lastly, the command is executed within the 3D Unity scene.

1.1 Contributions

Project members primarily contributed as follows.

- Joshua Jung integrated the major components of the project, including speech recording, speech to text translation, text to scene update translation, and real-time feedback in the VR headset. Joshua also engineered the initial versions of our LLM prompts.
- Michael Li designed the inference pipeline, including the overall structure and ordering of LLM prompts. Michael was also responsible for designing and constructing the relationship graph. Michael also provided background research and mostly wrote the project report.
- Eric Bae integrated Unity scenes with the VR headset and provided examples of the API for real-time scene updates. Eric also tuned the LLM prompts to optimize results.

2 RELATED WORK

2.1 Text-Conditional 3D Generation

Large Language Models (LLMs) are emerging as powerful tools for intuitive and dynamic 3D scene modification within virtual reality (VR) environments. LLMs can accurately infer human intent from natural language, and have shown success in image editing and generation in 2D [Fu et al. 2024; Wang et al. 2024; Zhang et al. 2023]. The natural next step would be extrapolating this functionality into the third dimension: by allowing LLMs to facilitate tasks ranging from object placement and manipulation to stylistic adjustments and even content generation in 3D, all by interpreting natural language commands. This technology would allow users to interact with VR scenes in a more natural and accessible way, eliminating the need for complex and technical interfaces, which often pose an accessibility barrier. There is prior work on generating 3D environments conditioned on text, but the resulting environments are often hard to dynamically modify [Shriram et al. 2025].

A major challenge to realizing 3D modification via natural language commands is linguistic ambiguity. Oftentimes, LLMs fail to interpret the intent of natural language commands [Liu et al. 2023; Ortega-Martín et al. 2023]. Sometimes, meaning is simply lost in translation and there is no plausible recovery. However, there are also many cases where when given enough context, it is possible to infer intent with reasonable accuracy. In our work, we seek to mitigate this problem for the latter mentioned case.

Additionally, current LLMs are not very good at reasoning in embodied or otherwise 3D environments. Current approaches for improving 3D reasoning often require special engineering or training, which make the resulting models less generalized [Hong et al. 2023; Zhen et al. 2024]. In practice, many developers do not have sufficient resources for training their own 3D reasoning models, and would prefer to handle 3D reasoning with API calls to general commercial LLMs such as ChatGPT, Claude, or Gemini.

2.2 Natural Language for 3D Manipulation

Prior work has touched on generating, modifying, and removing scenes and objects purely via textual descriptions. Chang et al. [2017]

propose SceneSeer, an interactive text to 3D scene generation system that allows a user to design 3D scenes using natural language. Kouzelis and Spantidi [2023] take scene generation into the realm of VR, being able to synthesize detailed, VR-ready 3D scenes from natural language. Their work utilizes a scalable database of 3D objects with minimal prior configuration. Bartrum et al. [2024] propose RAM3D, a novel method for 3D object replacement in 3D scenes based on text descriptions of the object to replace and the new object. Wang et al. [2025] propose VR Mover, which utilizes LLMs to interpret user vocal instruction for object manipulation. However, the manipulation supported is still relatively coarse grained, and do not support important practical features such as collision checking, concave insertions, or mid-air manipulation.

3 METHOD

Sentient Sandbox is an end-to-end system which takes spoken natural language commands as input, and then uses those commands to modify a 3D scene in real time. The pipeline initiates with the user speaking a command while viewing a 3D scene from a VR headset, powered by the Unity game engine. This audio input is first transcribed into text using the Whisper speech-to-text model. Next, the raw text is fed into the GPT-4o mini language model, which extracts structured instructions from the textual commands. Our system then maps these instructions to concrete actions supported by the Unity API. Currently, Sentient Sandbox supports object translation, scaling, and rotation. Before execution, we run a collision detection check to ensure that the intended action will not cause inconsistency within the scene. Finally, the validated command is executed within the Unity game engine, dynamically updating the 3D scene in real-time. Sentient Sandbox's architecture strives to provide a seamless interface between natural language input to precise 3D manipulation, enhancing accessibility and user experience.

The major algorithmic challenge that Sentient Sandbox tries to address is that of linguistic ambiguity: naively using large language models (LLMs) to directly extract technical instructions could result in inconsistent and ambiguous behavior. We use collision detection as a post-processing step to prevent inconsistencies in the resulting scene. However, this does not ensure that the edit actually aligns with the user's intent. Our key insight is to construct a relationship graph for the scene as a pre-processing step, and then provide this graph as context when interpreting the user prompt. The relationship graph explicitly encodes various properties of objects in the scene, as well as the relationships between different objects. Several of these properties are computed values, and not directly provided by Unity. Intuitively, the information provided by this graph simplifies the inference task for the language model, which results in greatly improved performance, especially in few-shot and zero-shot situations like our use case.

The architecture of Sentient Sandbox is depicted in Figure 2.

4 IMPLEMENTATION DETAILS

4.1 Relationship Graph

Upon scene initialization, we construct a relationship graph which encodes the properties of objects and their relationships with each other. The purpose of this graph is to explicitly provide important information about the scene as context to the language model, which simplifies the inference task and results in responses that better align with user intent. The relationship graph is undirected and contains no duplicate edges.

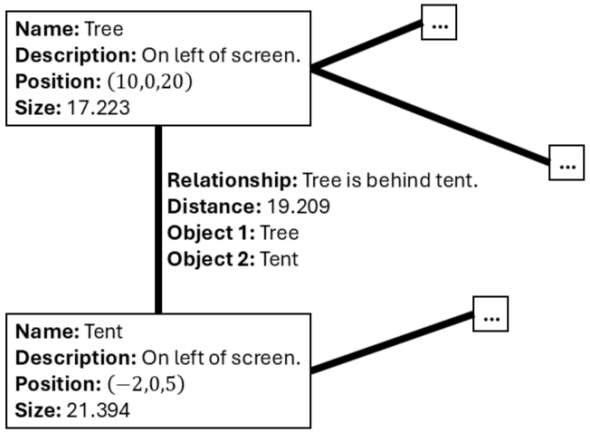

Fig. 3. Visualization of an example relationship graph. The relationship graph provides important context which improves the performance of LLM agents in extracting commands from user intent.

All objects in the scene are represented as nodes in the graph. The relationship graph cannot be fully connected because that would clutter our prompt with too much unnecessary information. We introduce an edge between two nodes if and only if they are within some distance d from each other. d is a hyperparameter which we manually set for different scenes. Both nodes and edges contain data, which is passed as context into the LLM prompt.

Nodes contain the following data fields:

- **Name.** The name is a manual label provided during scene design.
- **Description.** This field is currently used to specify the position of the object relative to the global frame.
- **Position.** This field is the (x, y, z) coordinates of the object, in camera coordinates.
- **Size.** Given a bounding box with dimensions (l, w, h), size is set to $\sqrt{l^2 + w^2 + h^2}$. This field is meant to provide a rough notion of size.

Edges contain the following data fields:

- **Relationship.** This field specifies the positional relationship between the two objects in natural language.
- **Distance.** Given object positions p_1 and p_2, this field is set to $\|p_1 - p_2\|$, the Euclidean distance between those points.
- **Obj 1 name.** This is the name of the first object that this edge connects to.
- **Obj 2 name.** This is the name of the second object that this edge connects to.

It is worth noting that object properties and relationships may change after scene modifications. Thus, after every scene modification, the affected data fields in the graph are recomputed.

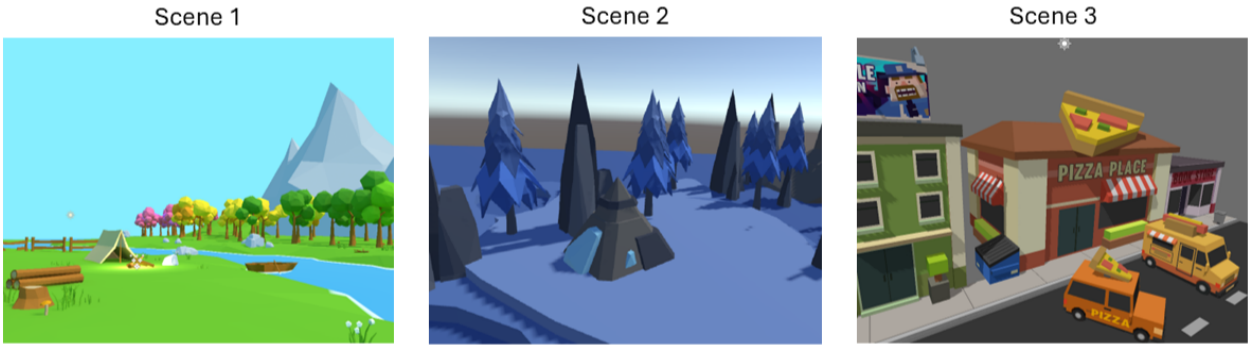

Fig. 4. Visualizations of the three scenes that we evaluate Sentient Sandbox on. These scenes vary in the shapes, sizes, and density of objects, and serve to test the robustness of our method in identifying and modifying precise objects.

It is also worth noting that several of these fields, most notably the size, distance, and relationship fields, are computed values. By providing this information in the context of our prompt, the LLM agent is able to more effectively reason about spatial relationships and provide an accurate interpretation of user intent.

Visualization of the components of an example relationship graph is provided in Figure 3.

4.2 Prompt Engineering

We engineer our prompt for GPT-4o mini with the objective of maximizing alignment with human intentions. The prompt that we use for extracting technical commands from user text has the following main components. The full prompt is rather long and is not included in this report for sake of brevity.

(1) Define the task. We start our prompt with the following command, to establish the role of the LLM.

> "You are an AI designed to process user commands for a VR scene using a scene graph. Your job is to analyze the scene graph, interpret the user's natural language instruction, and return a structured JSON response that includes the ID of the object being modified and the action to perform."

(2) Define rules. We specify the following rules for the LLM to follow when providing its response.

- Always respond with a JSON object that follows the exact format below.
- Do not add extra text, explanations, or comments. Only output the JSON.
- Ensure all values are properly formatted for execution in the‘ Unity engine.
- Use the scene graph to determine object relationships and select the correct object based on spatial context.
- If multiple objects match the description, choose the most relevant one based on distance and logical placement.
- If the request is ambiguous, make a reasonable assumption.
- If the action is "undo", simply return the action with an empty object ID.

(3) Define output format. We provide the JSON schema of the output that we expect.

(4) Provide examples. We provide an example relationship graph, as well as a few example commands to execute on it. We find that this improves performance. Intuitively, this makes sense because few-shot inference is almost always more robust than zero-shot inference.

(5) Provide context. We provide a string representation of the precomputed relationship graph. Note that in the prompt we refer to the relationship graph as the "scene graph".

4.3 Collision Detection

Collision detection is conducted by checking for overlaps between oriented bounding boxes (OBB). We choose to use OBB over axis-aligned bounding boxes (AABB) for improved performance, especially during rotations.

We use the Collider and Physics APIs from Unity to compute the bounding boxes and check for possible object collisions.

4.4 Technology Stack

We use the OpenAI Whisper API for speech to text conversion, and the OpenAI GPT-4o mini API for extracting commands from text.

Our 3D scenes are rendered with the Unity game engine, which exposes APIs for connecting to a VR headset and making real-time scene edits. Our demo scenes and their assets are retrieved from the Unity asset store.

We use a Meta Quest 3 as our VR headset. We primarily run our experiments on a Windows desktop with 32GB memory. However, we expect results to be reproducible on Windows machines with less memory, as well as on other operating systems.

5 EVALUATION OF RESULTS

5.1 Scene Experiments

We evaluate Sentient Sandbox on three different scenes, which vary in the shapes, sizes, and density of objects. These varied configurations serve to test the robustness of our method in accurately interpreting user intent and making precise modifications to the

scene. The three scenes that we evaluate Sentient Sandbox on are depicted in Figure 4.

Since Sentient Sandbox is an application, our evaluations are predominantly qualitative. We try out numerous prompts and find that our method behaves as expected in a majority of cases. However, we do experience several common failure cases that are worth noting:

- If there are multiple objects with the same name, such as multiple trees, our system struggles with identifying the correct object.
- Relationships between objects far from each other are not recognized, because graph edges are not created between objects that are more than d units apart.
- There is sometimes confusion between the "front" and "back" directions when relating objects.

5.2 Graph Ablation

We implemented our initial version of Sentient Sandbox without the relationship graph, and simply provided raw coordinates of objects within the prompt. This version was worse at interpreting human commands. For the most part, it performed reliably only when commands were phrased in a very particular way, to the extent that they sound unnatural.

5.3 Constraint Satisfaction

The only constraint violation we currently check for is object collision. In our experiments, we have not encountered cases where objects collide or otherwise end up on top of each other following scene modification. Thus, we are inclined believe that our collision detection mechanism is working properly with high probability.

5.4 Latency

The latency between finishing a verbal command and seeing the command get executed is around 4 seconds on average. In some cases, it can take as long as 6 seconds.

This latency is good enough for many practical applications where near-instantaneous updates are not required, such as design use cases. However, in other applications such as simulation use cases, this latency would be a hindrance and needs to be optimized.

6 DISCUSSION

6.1 Benefits

We engineered an end-to-end system for converting natural language commands to real-time 3D scene modifications in VR, which we are decently proud of, given our timeline. Our code could be open sourced and serve as a starting point for future works that use a similar setup.

We also demonstrated that using a relationship graph for prompt context improves performance, due to better spatial reasoning. This strategy could be replicated in future works of similar nature.

6.2 Limitations

This project operated under a significantly limiting time constraint, and many of our ideas on paper were not able to be implemented. Some immediate avenues for improvement include the following.

- We currently use a cutoff of d to determine whether an edge should be formed between two objects. Since d is a hyperparameter, it is prone to tuning errors, and also causes long distance relationships to be omitted. To mitigate this issue, we envision that methods from spectral graph theory can be used to better extract cluster properties based on distance relationships between objects.
- The latency we currently have is prohibiting for applications that require real-time responses. The delay can likely be optimized by improving prompting strategy for the text to command component, as well as using more eager evaluation for the speech to text component.
- We currently manually name objects during scene design. This part can be automated by running image recognition on the segmentation masks of individual objects. We did not implement this due to time constraints, and also because our wallets would be saddened by the number of API calls.

6.3 Future Work

We imagine that one avenue of future work will focus on enhancing the system's user interaction capabilities. A key improvement will be to allow real-time user navigation of the 3D environment, which is currently not supported. This would invoke engineering challenges, such as ensuring that spatial relationships and object properties are consistently accurate, regardless of the user's viewpoint changes. This will allow for a more immersive experience where users can intuitively explore and modify their surroundings, and make the system more suitable for a number of applications, especially as related to design.

Beyond immediate enhancements, future research will explore expanding the system's functionality and robustness. This includes developing more sophisticated methods for resolving linguistic ambiguity, such as incorporating contextual dialogue and user feedback mechanisms. Investigating the integration of more diverse object properties and relationship types into the relationship graph will also be beneficial, allowing for more nuanced modifications. Additionally, exploring the potential of integrating generative AI models for dynamic asset creation and style transfer could unlock new patterns for creative expression. Finally, optimizing the system's performance to minimize latency and improve scalability will be essential for deployment in real-time, high-demand applications.

7 CONCLUSION

In summary, Sentient Sandbox demonstrates a promising approach to real-time VR scene modification through natural language commands, leveraging a relationship graph to enhance LLM spatial reasoning. The system's strengths lie in its intuitive interface, robust collision detection, and improved interpretation of user intent. However, limitations in object identification, latency, and reliance on pre-defined scene elements highlight areas for future improvement. Addressing these challenges will be crucial for expanding the system's applicability to a wider range of dynamic and interactive VR experiences. Ultimately, this work contributes to the ongoing effort to create more accessible and intuitive 3D interaction paradigms.

REFERENCES

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Daniel Ho, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Eric Jang, Rosario Jauregui Ruano, Kyle Jeffrey, Sally Jesmonth, Nikhil J Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Kuang-Huei Lee, Sergey Levine, Yao Lu, Linda Luu, Carolina Parada, Peter Pastor, Jornell Quiambao, Kanishka Rao, Jarek Rettinghouse, Diego Reyes, Pierre Sermanet, Nicolas Sievers, Clayton Tan, Alexander Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Mengyuan Yan, and Andy Zeng. 2022. Do As I Can, Not As I Say: Grounding Language in Robotic Affordances. arXiv:2204.01691 [cs.RO] https://arxiv.org/abs/2204.01691

Edward Bartrum, Thu H Nguyen-Phuoc, Christopher Xie, Zhengqin Li, Numair Khan, Armen Avetisyan, Douglas Lanman, and Lei Xiao. 2024. ReplaceAnything3D: Text-Guided Object Replacement in 3D Scenes with Compositional Scene Representations. In *Advances in Neural Information Processing Systems*, A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (Eds.), Vol. 37. Curran Associates, Inc., 48568–48598. https://proceedings.neurips.cc/paper_files/paper/2024/file/571dd493cf500fc5bde887a6b5d4c941-Paper-Conference.pdf

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, Harsha Nori, Hamid Palangi, Marco Tulio Ribeiro, and Yi Zhang. 2023. Sparks of Artificial General Intelligence: Early experiments with GPT-4. arXiv:2303.12712 [cs.CL] https://arxiv.org/abs/2303.12712

Angel X. Chang, Mihail Eric, Manolis Savva, and Christopher D. Manning. 2017. SceneSeer: 3D Scene Design with Natural Language. arXiv:1703.00050 [cs.GR] https://arxiv.org/abs/1703.00050

Tsu-Jui Fu, Wenze Hu, Xianzhi Du, William Yang Wang, Yinfei Yang, and Zhe Gan. 2024. Guiding Instruction-based Image Editing via Multimodal Large Language Models. arXiv:2309.17102 [cs.CV] https://arxiv.org/abs/2309.17102

Yining Hong, Haoyu Zhen, Peihao Chen, Shuhong Zheng, Yilun Du, Zhenfang Chen, and Chuang Gan. 2023. 3D-LLM: Injecting the 3D World into Large Language Models. arXiv:2307.12981 [cs.CV] https://arxiv.org/abs/2307.12981

Lazaros Rafail Kouzelis and Ourania Spantidi. 2023. Synthesizing Play-Ready VR Scenes with Natural Language Prompts Through GPT API. In *Advances in Visual Computing*, George Bebis, Golnaz Ghiasi, Yi Fang, Andrei Sharf, Yue Dong, Chris Weaver, Zhicheng Leo, Joseph J. LaViola Jr., and Luv Kohli (Eds.). Springer Nature Switzerland, Cham, 15–26.

Alisa Liu, Zhaofeng Wu, Julian Michael, Alane Suhr, Peter West, Alexander Koller, Swabha Swayamdipta, Noah A. Smith, and Yejin Choi. 2023. We're Afraid Language Models Aren't Modeling Ambiguity. arXiv:2304.14399 [cs.CL] https://arxiv.org/abs/2304.14399

Miguel Ortega-Martín, Óscar García-Sierra, Alfonso Ardoiz, Jorge Álvarez, Juan Carlos Armenteros, and Adrián Alonso. 2023. Linguistic ambiguity analysis in ChatGPT. arXiv:2302.06426 [cs.CL] https://arxiv.org/abs/2302.06426

Jaidev Shriram, Alex Trevithick, Lingjie Liu, and Ravi Ramamoorthi. 2025. RealmDreamer: Text-Driven 3D Scene Generation with Inpainting and Depth Diffusion. arXiv:2404.07199 [cs.CV] https://arxiv.org/abs/2404.07199

Yanming Wan, Yue Wu, Yiping Wang, Jiayuan Mao, and Natasha Jaques. 2024. Infer Human's Intentions Before Following Natural Language Instructions. arXiv:2409.18073 [cs.AI] https://arxiv.org/abs/2409.18073

Xiangzhi Eric Wang, Zackary P. T. Sin, Ye Jia, Daniel Archer, Wynonna H. Y. Fong, Qing Li, and Chen Li. 2025. Can You Move These Over There? An LLM-based VR Mover for Supporting Object Manipulation. arXiv:2502.02201 [cs.HC] https://arxiv.org/abs/2502.02201

Zhenyu Wang, Aoxue Li, Zhenguo Li, and Xihui Liu. 2024. GenArtist: Multimodal LLM as an Agent for Unified Image Generation and Editing. arXiv:2407.05600 [cs.CV] https://arxiv.org/abs/2407.05600

Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. 2023. Adding Conditional Control to Text-to-Image Diffusion Models. arXiv:2302.05543 [cs.CV] https://arxiv.org/abs/2302.05543

Haoyu Zhen, Xiaowen Qiu, Peihao Chen, Jincheng Yang, Xin Yan, Yilun Du, Yining Hong, and Chuang Gan. 2024. 3D-VLA: A 3D Vision-Language-Action Generative World Model. arXiv:2403.09631 [cs.CV] https://arxiv.org/abs/2403.09631