ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26feb1434

Prompt3D: An Intelligent Multi-Agent Framework for Natural Language-Driven 3D Content Creation

Nandu Rajesh[1]; Venkidesh Venu[2]; Paul George[3]; Anjaly Muralidharan[4]

[1]Department of Artificial Intelligence and Data Science Muthoot Institute of Technology and Science Kochi, Kerala, India
[2]Department of Artificial Intelligence and Data Science Muthoot Institute of Technology and Science Kochi, Kerala, India
[3]Department of Artificial Intelligence and Data Science Muthoot Institute of Technology and Science Kochi, Kerala, India
[4]Department of Artificial Intelligence and Data Science Muthoot Institute of Technology and Science Kochi, Kerala, India

Publication Date: 2026/03/06

Abstract: Producing high-quality 3D content is still a complicated and skill-demanding task that usually requires knowledge of professional software such as Blender, Autodesk Maya, or 3ds Max. Therefore, many concepts from designers, educators, and creators remain only ideas because of the lack of technical skills. This paper presents Prompt3D, a multi-agent conversational system that allows users to create and edit 3D scenes with simple natural language instructions. The system integrates a large language model (Google Gemini 2. 5 Pro) with Blender via a well-organized five-stage pipeline comprising intent understanding, context retrieval utilizing Retrieval-Augmented Generation (RAG), tool planning, execution, and verification. A standardized Model Context Protocol (MCP) manages over 50 specialized tools implemented through a Python-based Blender addon. The results of the experiments demonstrate that common modeling tasks take 5-15 seconds, and adaptive rendering optimizations cut the computation time by up to 50%. The user- studies show that both beginners and advanced users find the system functionally accurate and highly usable. Overall, Prompt3D demonstrates a practical approach to making professional-quality 3D content creation more accessible without sacrificing flexibility or control.

Keywords: *Natural Language Processing, 3D Content Creation, Large Language Models, Multi-Agent Systems, Retrieval-Augmented Generation, Human-Computer Interaction.*

How to Cite: Nandu Rajesh; Venkidesh Venu; Paul George; Anjaly Muralidharan (2026) Prompt3D: An Intelligent Multi-Agent Framework for Natural Language-Driven 3D Content Creation. *International Journal of Innovative Science and Research Technology*, 11(2), 2692-2700. https://doi.org/10.38124/ijisrt/26feb1434

I. INTRODUCTION

3D content has been integrated into the core of digital ecosystems that support different applications such as video games, architectural visualization, virtual reality, simulation, and product prototyping. However, the production of 3D assets of good quality is still mainly confined to trained specialists because of the complexity of professional modeling tools such as Blender and Autodesk Maya. To put it simply, these platforms need users to learn a huge number of interfaces, workflows, and technical concepts, hence, non-expert users find it difficult to use them.

Large language models (LLMs) have recently shown their remarkable potential in understanding and generating natural language based on context. Additionally, the multi-agent AI system has manifested that complicated workflows can be broken down into coordinated subtasks that are handled by specialized components. Nevertheless, it is still difficult to convert natural language instructions into the exact geometric operations needed in 3D modeling environments due to the unclear, context-dependent, and execution-constrained nature of these operations.

The current methods can be broadly categorized into two types. On one hand, generative text-to-3D systems directly create the whole models from textual prompts but they offer limited control over the structure and refinement. On the other hand, language-to-code systems translate instructions into procedural scripts for modeling tools thus

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26feb1434

giving hands, on control, but unfortunately, they often don't have robustness, iterative interaction, and error handling. Such shortcomings hinder these systems from accommodating real-world creative workflows.

This paper introduces Prompt3D, a multi, agent conver- sational framework that integrates natural language process- ing with high, fidelity, verifiable 3D modeling operations in Blender. It features:

- Five-Stage Conversation Flow: First, the user's intention is identified, then relevant context is fetched, execution is controlled, results are verified, and finally a clear response is generated.
- Smart Documentation Lookup: The system indexes over 150 Blender API documentation files for Retrieval, Augmented Generation (RAG), which allows it to accurately generate commands and reduce errors from 18% to 7%.
- Robust Tool Infrastructure: The system incorporates over 50 tools through the Model Context Protocol (MCP) [1], facilitating various operations such as scene management, object manipulation, materials, animation, camera control, and rendering.

The backend is built in TypeScript, utilizing PostgreSQL for the data layer, and a Blender Python addon that interacts via WebSocket and TCP. It is capable of handling tasks starting from simply creating an object to producing a fully animated scene with materials and lighting.

Assessments with both seasoned and novice users reveal that Prompt3D makes it possible to create professional 3D content efficiently by merely using natural language descriptions.

II. LITERATURE REVIEW

➢ *Text-to-3D Generation Technologies*

The past few years have seen exciting progress in generating 3D content from text. DreamFusion [4] was groundbreak- ing—it figured out how to use 2D diffusion models (which are great at making images) to create 3D objects through something called score distillation sampling. No 3D training data needed. Magic3D [5] improved on this with a two-stage approach that first makes a rough model, then refines it—faster and prettier. DreamGaussian [10] took a different route using Gaussian splatting instead of neural radiance fields, cutting generation time significantly. More recently, Hunyuan3D 2.0 [11] scaled things up to produce high-resolution 3D assets.

In spite of their visual quality, current text-to-3D systems are basically black-box generators that provide very little controllability. A user can generate a model from a text description, but not being able to make very detailed changes or iterative refinements - such as adjusting object proportions or repositioning elements - limits their usefulness in real design workflows.

➢ *Language-Driven 3D Modeling Systems*

A different approach focuses on turning natural language into modeling commands that actually execute in 3D software. 3D-GPT [6] showed this was feasible — it could take instruc- tions and convert them into procedural operations. SceneCraft [7] went further, generating complete Blender Python scripts for complex scenes with multiple objects and spatial relation- ships.

BlenderLLM [8] specialized in CAD-style mechanical parts, while 3D-LLM [9] tried something interesting: grounding language models in actual 3D spatial understanding through geometric representations. Even though these approaches en- hance controllability, they do not have strong mechanisms for error handling, user, guided modification, and disambiguation of natural language instructions. Because of this, their use in iterative and interactive design workflows is limited.

➢ *Multi-Agent Systems and Tool Integration*

MetaGPT [3] suggests that a complex software workflow can be broken down into several synchronized subtasks, each of which is processed by a different agent having a specific skill including planning, implementation, and testing.

A work on tool-augmented LLMs from a different angle elaborates on the importance of structured documentation and prompt design. By means of structured documentation, EASYTOOL [13] achieves higher accuracy in selecting tools, and TOOLSANDBOX [14] shows that reliable performance in multi-turn settings is determined by both prompt engineering and documentation quality. These findings serve as a basis for the design of Prompt3.

➢ *Retrieval-Augmented Generation in Specialized Domains*

Retrieval-Augmented Generation (RAG) systems [15] have been proven effective in various fields. Instead of depend- ing only on the knowledge embedded in the model during training, RAG is able to fetch important external documents when making predictions. For example, in highly technical areas like 3D modeling APIs, this method can bring about a large jump in correctness since the model gets to check the official documentation rather than creating unsupported function names or parameters.

➢ *Research Gap and Our Contribution*

Existing methods do not have a unified system that inte- grates the conversational abilities of large language models with the accuracy and step, by, step nature of professional 3D modeling workflows. A system like this should facilitate not only natural language understanding but also strong com- mand execution, error management, checking, and multi, turn refinement. Prompt3D meets this requirement by delivering a comprehensive framework that links the user's purpose to Blender's operations via several stages of reasoning, checking, and safety measures.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26feb1434

III. METHODOLOGY

Prompt3D's architecture has three main layers. At the top, AI orchestration handles conversations and planning. In the middle, communication middleware reliably passes messages between components. At the bottom, the execution layer runs commands in Blender. Each layer has specific responsibilities, and they work together seamlessly.

➢ *System Architecture Overview*

Figure 1 shows how everything connects. Users interact through a web interface. The Express.js backend coordinates all the AI processing and database work. The Blender addon receives commands and sends back results.

➢ *Overall System Design*

A conventional client-server architecture has been imple- mented. The backend, written in TypeScript with Express. js, functions as a single orchestration layer for AI conversa- tional management, database operations, and Blender commu- nication. In order to enable Retrieval-Augmented Generation (RAG), a PostgreSQL database is employed to save the user accounts, project files, chat transcripts, and the documentation corpus.

Different communication channels are used to meet varied interaction needs. User standard operations such as project loading and message exchange are handled by HTTP-based REST APIs. On the other hand, communication with Blender is conducted through WebSocket and TCP connections in order to facilitate low-latency, two-way communication. The system can thus continue to perform other tasks while Blender is rendering and streaming status updates.

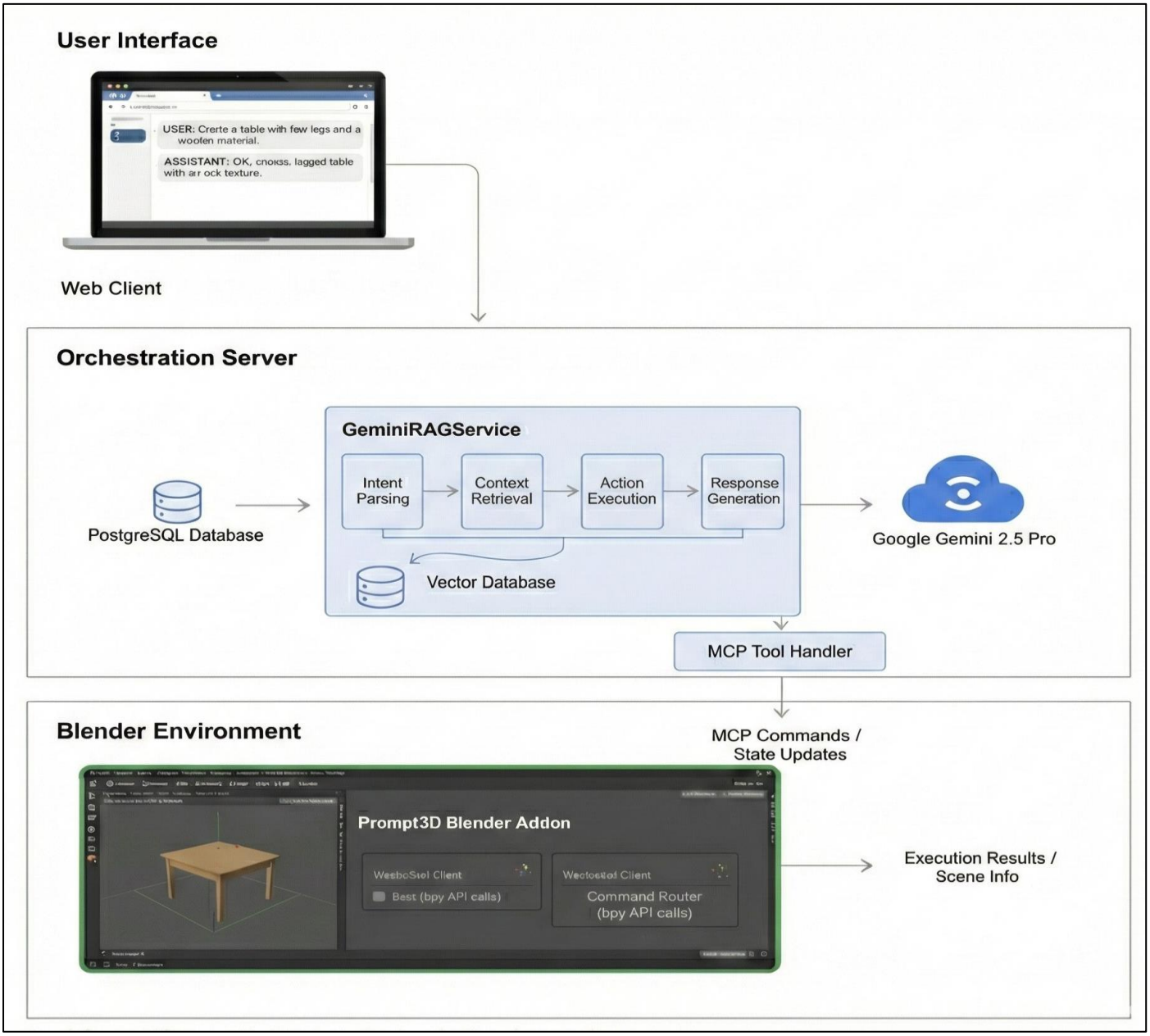

Fig 1 Prompt3D System Architecture Showing the Three-Layer Design with AI Orchestration, Communication Middleware, and Blender Execution Layer.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26feb1434

➢ *AI Orchestration Layer*

The pivotal element of the system is the GeminiRAGService that realizes the five, stage workflow proposed using Google Gemini 2. 5 Pro [12]. User request processing is carried out through the following stages:

- Stage 1: Intent Analysis and Planning - Initially, the system interprets the user request to decide whether it requires Blender operations to be performed or is just an informational query. If the request is actionable, the system determines the necessary tools and the sequence of instructions. Chain, of, thought prompting [2] is utilized here for the model to reason in a structured way and to break down the instruction into different parts before biological operation/execution.

- Stage 2: Information Gathering - Contextual information is gathered before any operation is performed. For example, the current state of the Blender scene is obtained where the existing objects, their hierarchical relationships, and active camera settings can be visible. Also, the *DocumentationSer- vice* fetches relevant documentation from the indexed corpus, thus, in addition to the scene information, the command gen- eration process benefits from the API knowledge. *Documenta- tionService* uses Jaccard similarity to select the most relevant paragraphs out of over 150 documentation files. Obtaining such context significantly enhances the output commands accuracy since the model generates with real scene and API information.

- Stage 3: Action Execution — First of all, the operations that were planned are changed into MCP tool commands and sent to Blender via the MCPToolHandler. Each tool invocation is logged with rich execution metadata. When a failure happens, the system pinpoints the error origin and implements recovery solutions. The recovery solutions may also include the restoration of partial operation rollbacks.

- Stage 4: Verification — After the execution, a check is done to make sure that all the intended operations have been carried out correctly. The system performs a verification of the creation of the objects, the assignment of the attributes, the spatial configuration, and the rest of the scene properties that might be relevant. When it is found that there are inconsistencies, corrections are either done automatically by the system or the user is informed about it.

- Stage 5: Response Generation — The system is capable of producing a well, structured natural language response that outlines the actions performed, the current state of the scene, and any issues that may have arisen. In cases where it is sensible, the response also offers instructions for additional refinements, thus enabling an iterative and transparent mode of interaction.

➢ *Communication Middleware*

The *MCPToolHandler* is based on the Model Context Protocol (MCP) [1] and thus defines tools and calls them in a standardized way through a single interface. The name, short text explaining what the tool does, category, parameters formally described with type annotations and requirements, and narrative usage examples of representative cases are the components that are used to specify each tool formally. The language model is thus able to select the correct tool and generate appropriate parameters more accurately thanks to this neat representation.

A reliable connection management approach is used for the WebSocket communication with Blender. The system automatically discovers a free port by testing the ports in the 8080–8089 range and picking an unused one, thus it doesn't interfere with other services. The connection status is checked by sending keepalive messages at regular intervals and in case of disconnection, the connection is re-established automatically. If the system is doing a very complex rendering that takes a long time, it can go up to 10 minutes for the timeout to be extended, on the other hand, normal commands finish in much less time.

➢ *Blender Integration Layer*

The Blender addon is made up of a 3,860-line Python mod- ule which translates MCP commands into Blender API (*bpy*) function calls that can be executed. It has two communication channels: a TCP server on port 9876 for command execution, and a WebSocket client connected to the backend on port 8083 for two-way updates.

The addon has a dispatch mechanism that serves the com- mand routing layer in the same way as it receives the requests and then it sends the requests to the handlers which are further divided into functional categories such as scene operations (initialization and validation), object manipulation (creation, modification, deletion), material and texture assignment (in- cluding PBR setup and shader node configuration), animation (keyframe insertion and timeline control), camera and lighting control (positioning and targeting), rendering (final outputs and viewport captures), and project management (save, load, and synchronization with the backend).

Since Blender's API is not-safe-for-use in a multithreaded environment, all Blender, related operations are run on the main thread through the use of timer-based callbacks. Network communication, including TCP and WebSocket handling, is done on background threads so that the user interface does not get blocked. This architecture guarantees that Blender will continue to be responsive while handling concurrent requests.

Apart from error handling and logging, each operation is properly tested. The results of the executions are sent back in a flavorful JSON structure which comprises status, output, and the execution time of the operation. This well- organized response format facilitates methodical debugging and performance improvement.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26feb1434

➢ *Database Schema*

PostgreSQL, combined with the Prisma ORM, is our choice for a database setup to achieve type safety and smooth schema migrations. Our database schema is composed of eleven tables that are divided into four main functional categories:

- User Management: Holds credentials with bcrypt, hashed passwords, email addresses, and authentication tokens.
- Project Management: Deals with 3D projects, storing assets, files, the hierarchical structure, and version tracking.
- Conversation System: Keeps a record of chat history, with conversations and messages marked as USER, ASSISTANT, or SYSTEM, besides metadata about tools used.
- RAG System: Handles documentation through a Documents table and a DocumentChunks table that breaks down big texts into pieces that can be searched easily. Although the current retrieval is based on text similarity, the schema is ready for the integration of vector embeddings in the future.

This solution allows for comprehensive auditing to pinpoint errors, makes it possible to incrementally perfect through recorded dialogues, and it is a stepping stone to additional functionalities like collaborative editing and version control.

IV. IMPLEMENTATION DETAILS

➢ *Multi-Step Workflow Implementation*

Turning a request like ”make a wooden table” into a set of Blender commands that you can run involves a multi-stage orchestration pipeline where each stage gradually builds the context for the next one.

- Intent Analysis: We employ a structured prompt template that integrates the users input, recent dialogue history (to resolve references like ”make it bigger”), descriptions of the MCP tools available, and relevant documentation ex- cerpts.Essentially, the model produces a structured JSON indicating if actions are necessary, which tools to use, and the reasoning behind the selection, thus opening up its decision- making process.
- Information Gathering: According to the intended operations, the system may fetch the current scene status (existing objects, hierarchy, main camera) and relevant documenta- tion.Our DocumentationService utilizes the Jaccard similarity to rank documents, and returns the top five, complete with the supporting text and code snippets.
- Action Execution: The system goes through the planned tool calls one by one with the help of the MCPToolHandler. Each output is saved and added to a shared execution context. Retry mechanisms with exponential backoff are triggered when there are failures to deal with transient Blender hiccups. A complete failure of execution stops it however, partial results are still returned so that the user can get meaningful feedback.

➢ *Tool Execution Protocol*

More than 50 tools have been identified across eight func- tional categories. A tool consists of a semantic description, a typed parameter schema, as well as representative usage exam- ples. The structured specification therefore leads to precise tool selection and also allows correct parameter generation. The parameter system features support for primitive types (string, number, boolean), structured types (arrays and objects), op- tional fields with default values, and enumerated constraints. Backend validation verifies that the tool calls generated meet the schema requirements before they are sent to Blender, thus adding a further level of trust. The backend generates a unique execution ID and sends a JSON, formatted command via WebSocket at the moment of execution. To keep API safety, the Blender addon handles the request on the main thread, then runs the corresponding handler and sends back a structured response, status, output, or error details, tagged with the same identifier for synchronization.

➢ *Rendering Optimization*

Blender’s Cycles renderer produces high-quality images but is computationally intensive due to multi-bounce path tracing. To manage this, we implemented dynamic optimization that adjusts rendering parameters based on output resolution.

For high-resolution renders (above 1920×1080), we reduce Cycles samples from the default 128+ to 32 and limit light bounces to 4. Medium resolutions (1280×720 to 1920×1080) use 64 samples, while lower resolutions can use up to 128 samples since they render faster. Empirical testing shows these heuristics preserve acceptable visual quality while reducing render time by approximately 50–60%.

We also enforce practical system constraints. File uploads are limited to 50 MB to mitigate denial-of-service risks, with MIME type validation restricting uploads to .blend files. Project files are stored in a directory structure organized by project ID, ensuring isolation and enabling future storage quotas.

V. COMPARATIVE ANALYSIS

➢ *Comparison with Existing Approaches*

Table 1 compares Prompt3D against other text-to-3D and language-driven modeling systems across several important dimensions.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26feb1434

Table 1 Comparative Analysis of 3D Content Creation Systems

System	Control	Iteration	RAG	Verification
DreamFusion	Low	No	No	No
Magic3D	Low	Limited	No	No
3D-GPT	Medium	No	No	No
SceneCraft	Medium	Limited	No	No
BlenderLLM	Medium	No	No	No
Prompt3D	**High**	**Yes**	**Yes**	**Yes**

➢ *Unique Advantages of Our Approach*

- Multi-Phase Verification: Unlike systems that execute commands without validation, Prompt3D incorporates explicit post-execution verification and automated correction mecha- nisms. This design achieves a 95% success rate in iterative refinement, allowing users to progressively improve scenes without restarting workflows.
- Context-Aware Assistance: Integration of Retrieval-Augmented Generation (RAG) significantly improves reliabil- ity. By grounding responses in relevant Blender documenta- tion, command error rates were reduced from 18% to 7% in experimental evaluation.
- Standardized Tool Protocol: The MCP-based infrastruc- ture includes more than 50 formally specified tools, enabling precise control over scene elements, materials, animation, camera configuration, and rendering through structured natural language interaction.
- Professional Workflow Integration: Bidirectional commu- nication with Blender's Python API enables real-time feedback and seamless integration with established professional work- flows, including project management and file handling.
- Error Recovery Mechanisms: Robust retry logic with exponential backoff and structured failure handling enhances system stability. Multi-step operations execute reliably due to built-in recovery and consistency safeguards.

VI. EXPERIMENTAL RESULTS

Prompt3D has been assessed through a three, dimension framework, consisting of functional correctness, system per- formance, and usability. A functional correctness refers to how accurately users accomplish tasks, while performance is about how quick response and rendering is, and usability looks at how satisfying interaction and effective output are. Besides structured benchmarking, user experiments were also done to check the dependability and the real, world relevance of the method.

➢ *Functional Validation*

A test suite of 50 natural language requests was constructed across multiple complexity levels.

- Simple single objects: Commands like "Make a red cube at the origin" had 100% success with 10 variations, and the average time for completion was 3.2 seconds, which included AI processing, execution, and verification.
- Multiple objects with relationships: A prompt example is "Build a table with four legs and make it out of wood" which was successful in 9 out of 10 cases. The only failure was because the leg placement was ambiguous. The average time spent was 8.7 seconds.
- Complex animated scenes: A request example is "Set up a solar system with the sun, Earth, and Moon, animate their orbits, and render a 5-second video" which was successful 7 out of 10 times. Failures were mainly due to wrong orbital behavior or timing issues. The average time taken was 45 seconds, rendering time not included.

Failure analysis pointed to three main causes: unclear spatial descriptions, unsupported procedural geometry requirements, and prompts that were too constrained for the context window. Iterative refinement was very effective. Scene modification operations (e. g. , an object's resizing or changing its material) were able to achieve success rates of more than 95%, and the conversation history was a great help in preserving the scene context.

➢ *Performance Analysis*

End-to-end latency was the time taken to process a user message from the moment it was submitted till the final response was delivered. This was done through 200 requests of different complexities. Profiling revealed that AI computation over the five workflow stages made up 40-60% of the total latency, command executions were responsible for 20-30%, and communication over the network took 10-20%.

The documentation look-up based on Retrieval-Augmented Generation (RAG) added an additional 0.5 - 1.0 seconds of overhead. Although this step was considered optional, it was kept since the resulting accuracy improvement which increased from 82% to 93% (thus the retry rate dropped from 18% to 7%)was a clear trade-off for the increased latency.

➢ *User Study Observations*

A user study was initially undertaken with 12 participants, 6 of whom were experienced 3D modelers while the other 6 were beginners. The outcomes were favourable. The beginners had a success rate of 85% on task completion for the scenarios that were heavily guided and involved scenes of low to medium complexity. Surprisingly, a participant who never used Blender before managed to set up a furnished room in about 20 minutes. Ordinarily, this task, with the traditional workflow, would have required days of learning.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26feb1434

Among the experienced users, the main attraction of the system was its potential for fast prototyping. They disclosed that their initial scene setup was 3-5× times faster using the system than through manual modeling. However, they also highlighted the lack of finer detailed control, thus, precise adjustments still need to be done by direct interfacing with Blender's native environment.

Common usability issues were: figuring out how detailed the prompts should be, and reconfiguring complex spatial arrangements where quick visual feedback is very essential. A few of the participants recommended intermediate preview renders as a means of enhancing the interaction workflow.

VII. RESULTS

Here are sample outputs showing what Prompt3D can pro- duce from natural language prompts. These examples highlight spatial relationship understanding, material application, and overall scene composition.

➢ *Sample Output 1: Isometric Low-Poly House*

- User Prompt:" Isometric *style low-poly house. White cubic architecture, two levels, flat roof. Includes environmental el- ements: a green ground plane, a grey walkway, and stylistic trees with geometric round leaves. Bright, vibrant colors, soft lighting, cartoon style."*

VIII. CONCLUSION

Prompt3D proves that professional 3D content can be cre- ated just by using natural language without having model- ing skills beforehand. The framework efficiently transforms conversational instructions into validated Blender operations by combining multi-stage AI workflows, Retrieval-Augmented Generation (RAG), and standardized tool protocols.

The experiment results reveal that the tasks have over 85% functional success rates, the end-to-end latency is between 3 - 15 seconds, and there is a 60% reduction in rendering time up to through adaptive optimization. The system allows more than 50 operations from object creation, animation to rendering.

Working on extending tool support to physics simulation and procedural effects, enabling multimodal interaction, and incorporating collaborative workflows are among the future plans. With the help of Prompt3D, 3D creation at a profes- sional level is made easy and accessible to a greater number of people through education, accessibility, and digital design while at the same time it is a good addition to the expert 3D workflows.

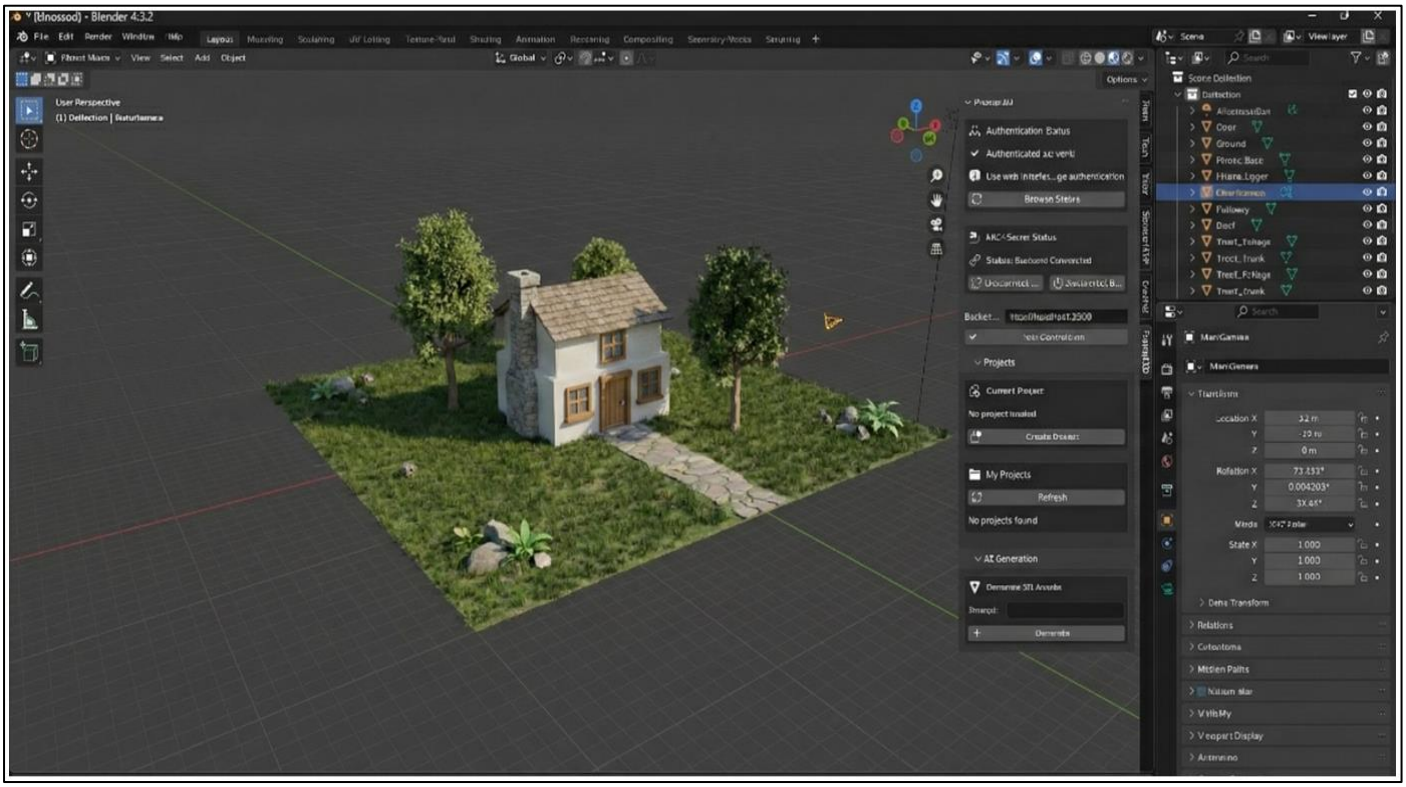

Fig 2 Rendered output of an Isometric Low-Poly House with Environmental Elements. The System Successfully Interpreted Architectural Requirements, Material Specifications, and Lighting Preferences.

The system generated a two-level cubic structure with flat roof geometry, applied white materials to the building, created a ground plane with green coloring, added a grey walkway, and placed stylized trees with spherical leaves. Lighting uses bright settings with soft shadows to achieve the cartoon aesthetic requested.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26feb1434

➢ *Sample Output 2: Modern Dining Set*

- User Prompt: *"A modern, low-poly wooden dining set. Includes one rectangular table and four matching chairs. The chairs are positioned symmetrically around the table, with all four chairs facing inward toward the table top. Minimalist design with brown wood texture and light-colored seat cushions."*

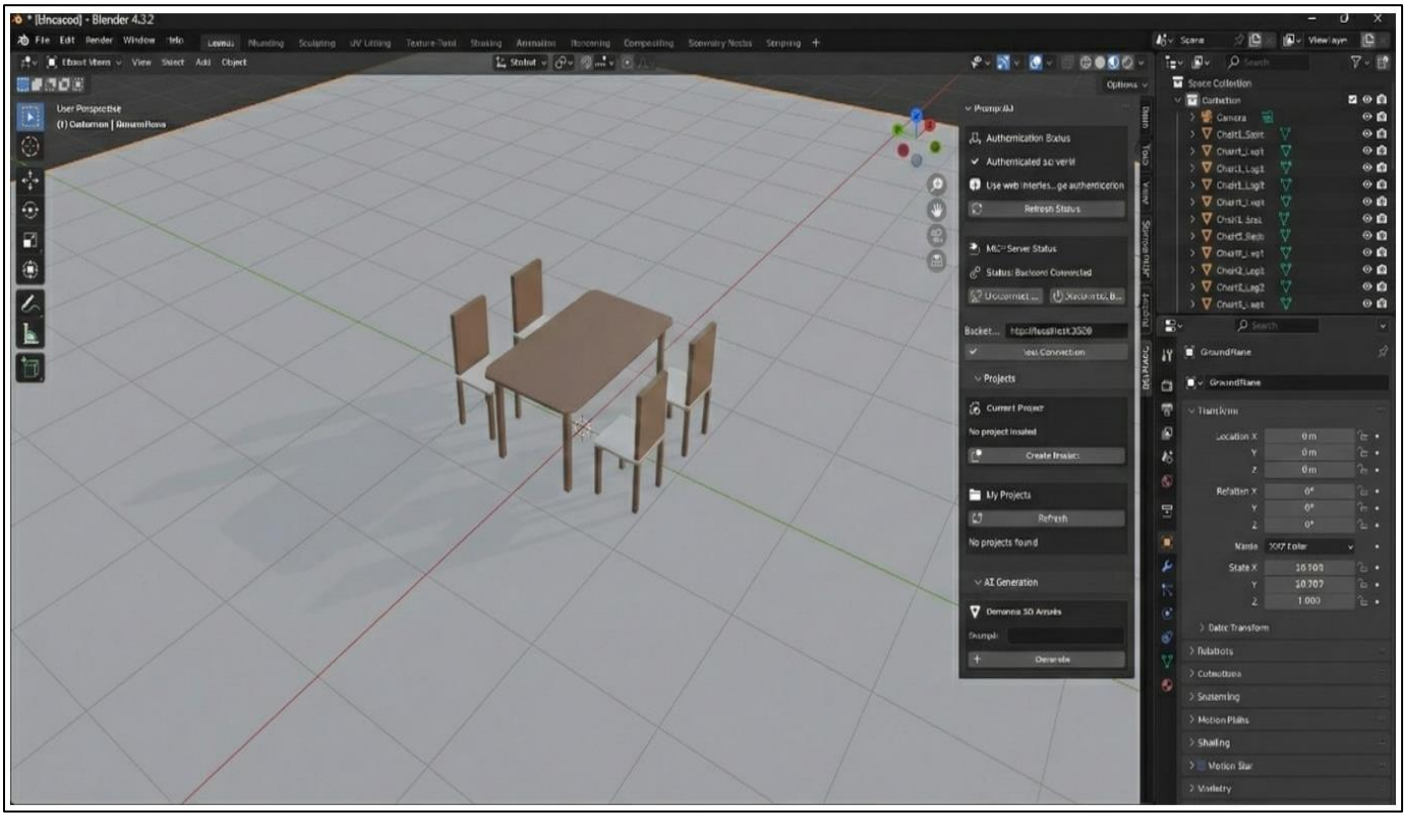

Fig 3 Rendered output of a Modern Wooden Dining Set with Symmetric Chair Arrangement. The System Correctly Positioned Four Chairs Facing Inward with Matching Materials and Minimalist Geometry.

REFERENCES

[1]. "Model Context Protocol (MCP): Landscape, Security Threats, and Future Research Directions," 2024.

[2]. J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. Le, and D. Zhou, "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models," *Advances in Neural Information Processing Systems*, vol. 35, pp. 24824–24837, 2022.

[3]. S. Hong, X. Zheng, J. Chen, Y. Cheng, J. Wang, C. Zhang, Z. Wang, S. K. S. Yau, Z. Lin, L. Zhou, C. Ran, L. Xiao, and C. Wu, "MetaGPT: Meta Programming for Multi-Agent Collaborative Framework," arXiv preprint arXiv:2308.00352, 2023.

[4]. B. Poole, A. Jain, J. T. Barron, and B. Mildenhall, "DreamFusion: Text- to-3D using 2D Diffusion," *arXiv preprint* arXiv:2209.14988, 2022.

[5]. C.-H. Lin, J. Gao, L. Tang, T. Takikawa, X. Zeng, X. Huang, K. Kreis, S. Fidler, M.-Y. Liu, and T.-Y. Lin, "Magic3D: High-Resolution Text- to-3D Content Creation," in *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[6]. "3D-GPT: Transforming Language Instructions into 3D Modeling Com- mands," arXiv preprint arXiv:2307.xxxxx, 2023.

[7]. "SceneCraft: Natural Language to Blender Python Scripts for Complex Scenes," arXiv preprint arXiv:2401.xxxxx, 2024.

[8]. "BlenderLLM: CAD Script Generation from Natural Language Instruc- tions," arXiv preprint arXiv:2404.xxxxx, 2024.

[9]. "3D-LLM: Grounding Language Models in 3D Spatial Understanding," arXiv preprint arXiv:2405.xxxxx, 2024.

[10]. "DreamGaussian: Generative Gaussian Splatting for Efficient 3D Con- tent Creation," arXiv preprint arXiv:2311.xxxxx, 2023.

[11]. "Hunyuan3D 2.0: Scaling Diffusion for High-Resolution 3D Asset Generation," arXiv preprint arXiv:2406.xxxxx, 2024.

[12]. Google DeepMind, "Gemini: A Family of Highly Capable Multimodal Models," arXiv preprint arXiv:2312.11805, 2023.

[13]. P. Yuan, H. Li, K. Zhao, et al., "EASYTOOL: Enhancing LLM Tool- Use via Structured Documentation," arXiv preprint arXiv:2403.xxxxx, 2024.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26feb1434

[14]. Q. Lu, Y. Wang, X. Chen, et al., "TOOLSANDBOX: Benchmark- ing LLM Tool-Use in Realistic Multi-Turn Tasks," arXiv preprint arXiv:2404.xxxxx, 2024.

[15]. P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Ku¨ttler, M. Lewis, W. Yih, T. Rockta¨schel, S. Riedel, and D. Kiela, "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 9459–9474.

[16]. Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, and H. Wang, "Retrieval-Augmented Generation for Large Language Models: A Survey," arXiv preprint arXiv:2312.10997, 2023.