Published by :
https://www.ijert.org/
An International Peer-Reviewed Journal

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

Prompt2Model: An MCP-Powered AI System for Converting Natural Language Prompts into 3D Models

Dr. N. Sree Divya
Department of Information Technology
Mahatma Gandhi Institute of Technology

Uppala Rohith
Department of Information Technology
Mahatma Gandhi Institute of Technology

Dhanush Ravala
Department of Information Technology
Mahatma Gandhi Institute of Technology

Abstract - Object creation in the 3D space requires several procedures. These include object set up, texturing, object importing, and light source introduction. However, the above procedures necessitate input from a beginner due to some challenges associated with 3D modeling software used by experts. Therefore, Prompt2Model offers solutions to this problem by automatically translating English prompts to geometry and textures through Blender.

Notably, it is imperative to point out that this system employs two phases. Firstly, this system uses the Claude language model from Anthropic, whereas the second phase deals with developing a Blender plugin for interacting with Blender using Model Context Protocol. The reason behind using this approach in creating a 3D system is that the approach includes incorporating external sources, including Polyhaven for texturing and HDRI, Hyper3D Rodin for building meshes, and Sketchfab for finding 3D models. Therefore, it is possible to build a complete scene based on the user's natural language prompt.

Keywords: Text-to-3D; Blender Automation; Model Context Protocol; Large Language Models; Generative AI; Procedural 3D Modeling; AI-Assisted Content Creation

1. INTRODUCTION

The difficulty that one has to address before embarking on the modeling process is extremely challenging, even up to this day.Programs like Blender, Maya, and 3ds Max are very robust, yet their steep learning curve makes them too complex for the average individual willing to invest years learning how to use them.This situation is particularly frustrating because the demand for 3D content has never been higher particularly game developers, architects, educators, filmmakers, and XR designers all need it, and that pool of people who "just want a 3D object" extends far beyond the group of people willing to spend months learning viewport navigation shortcuts.

The recent development in the large language models space paved the way to the possibility of turning natural language instructions into executable instructions. Turning natural language instructions into executable instructions is no longer an option but rather a necessity when it comes to building agentic systems. We built Prompt2Model to explore what that idea looks like when applied to 3D creation: specifically, to the question of whether a user could describe a scene in conversational language and have a system construct it, start to finish, inside a real professional 3D environment.

We designed Prompt2Model to use Blender not merely as a software application but as an orchestration platform.Normally, text-to-3D systems produce either unconnected meshes or
2.5D images.In our case, the objective was to make use of Blender to control meshing operations, assign cycle-compatible materials, load third-party models, create cameras, and perform lighting in accordance with natural language prompts. This is where our Model Context Protocol proved helpful: it provided us with a simple yet elegant mechanism of doing what needed to be done through an API via which the language model calls function available through a third-party software, whereby, in our case, the latter was a TCP socket server running inside Blender.Below there will follow several subsections devoted to the presentation of background of our work (Section 2), problem description (Section 3), architecture of the approach we developed (Section 4), implementation details (Section 5) and experimental results (Section 6).

2. RELATED WORK

2.1.Neural 3D Representation and Generation

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

The methodologies utilizing learning algorithms for producing three-dimensional images showed substantial advancements in the past five years.Although the first techniques relied on voxels, point clouds, and mesh auto encoders, relatively simple models yet quite inefficient for complex representations, the breakthrough technology known as NeRF [1] demonstrated that implicit volumetric models could be trained on just several views of a photorealistic scene. In particular, the reconstruction problem attracted researchers' attention for some time, but recently notable progress has been made due to DreamFusion [2], where SDS sampling enabled training of a neural radiance field using optimization of such a network to generate an object solely according to a text prompt without any exposure to 3D samples. Follow-up research included various works, namely Magic3D [3], Fantasia3D [4], and ProlificDreamer, aimed at increasing efficiency of neural networks, whereas MVDiffusion and Zero-1-to-3 focused on multi- view consistency as a key concept behind accurate 3D shape generation. As far as mesh generation is concerned, PointE and ShapE [5] created by

OpenAI proved transformers' ability to rapidly generate point clouds and even implicit models for 3D objects based on large amounts of 3D information with a focus on performance rather than quality.Hyper3D Rodin is yet another architectural design that could be employed in the existing scenario, whereby diffusion models would be used to generate mesh networks based on the input prompt or image. Rodin is among the modules present in our orchestration framework, but the major difference lies in the fact that the whole architecture has been developed around it.

2.2.LLM-Driven Code and Tool Use

Commercial tools such as Spline AI and Kaedim support text-driven mesh generation for architectural visualization.More structured approaches like ToolFormer and HuggingGPT explored teaching models to select and call external tools via API, while the OpenAI function-calling mechanism and Anthropic's tool use capability formalized this pattern into a stable interface. The Model Context Protocol, introduced more recently, extends the idea by defining a standard wire protocol through which a host application can expose tool definitions to an LLM, receive structured tool calls, and return results — without either side needing to know the internal implementation details of the other.

BlenderMCP, the open-source project we built on top of, was one of the first demonstrations of MCP applied to creative software. It established the core pattern of running a TCP socket server inside Blender as an add-on and connecting it to an MCP-capable LLM host, allowing the LLM to issue commands like "create a cube at position (0, 0, 1)" or "apply a red metallic material to the selected object." Our contribution extends this foundation into a full scene-building pipeline that spans multiple external services and handles more complex compositional tasks.

2.3.AI-Assisted 3D Content Creation

Several commercial and research systems have explored AI assistance in 3D workflows. Adobe's Firefly has been extended toward 3D asset generation, and NVIDIA's GET3D and Magic3D represent academic efforts at high-quality single- object synthesis. In terms of architectural visualization, there is software such as Spline AI and Kaedim that uses text-to-mesh and image-to- mesh approaches, although these tools work in closed environments. What distinguishes Prompt2Model is not any single capability in isolation, but the integration of an LLM as an orchestration layer that decides, based on semantic understanding of the user's intent, which combination of tools and services to invoke and then executes that plan directly within the user's existing Blender project.

3. PROBLEM STATEMENT

What the system is intended to do is quite straightforward, but at the same time, very difficult to do: using a textual description of a scene in plain English, generate the corresponding scene/object in Blender without the user being required to write any computer program code or select anything from a drop-down list, or even know anything about 3D modeling software tools. There are certain issues with accomplishing this task, however. Firstly, there are features in generating algorithms for processing the plain English scene description and implementing the corresponding functionality that have nothing to do with the actual generation of the scene itself. Thus, although we understand that the room is comfortable, we cannot assume that its size, location, or even material composition are known to us, or the source of light within. Secondly, the environment where our system is going to operate presents a serious challenge, too. It needs to be able to run on Python within the Blender environment; unfortunately, it lacks sufficient robustness to ensure its survival through any function call that could possibly cause a crash of the entire software. Thirdly, the third issue that should be addressed involves the creation of art based on different information sources. With the help of LLM, one can expect an understanding of semantics as well as sensible defaults to fill in some parts of prompt input. The request protocol utilizes JSON-based structures instead of code generation, making the list of possible responses

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

limited to commands executable by the plug-in.

4. SYSTEM ARCHITECTURE

The Prompt2Model architecture has four major modules that include:

• Blender plug-in acting as a TCP socket server,

- MCP Client having Claude module for generating prompts and tools commands,
- Command handler in the Blender plug-in managing the commands and distributing them among different handlers, and

• Integration Plug-ins invoking the APIs.

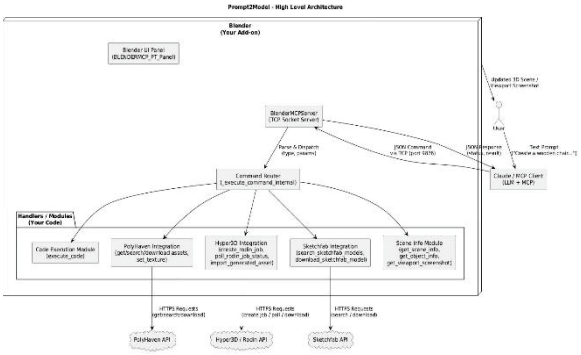


Figure 1: Structure of the Prompt2Model software application showing the way how the prompts are sent to the software application until reaching the Claude/MCP client, TCP socket server, command dispatcher, and integrating third-party services.

4.1.The Blender Add-on and TCP Server

The plug-in is embedded within the user interface of the graphical program Blender, and when it is launched, it launches the TCP socket server BlenderMCPServer. This server operates consistently, passing messages containing commands and their parameters to the command dispatcher. Information included in such messages may pertain to the results or data from the process execution in the form of JSON objects. Socket server implementation of TCP using the Blender user interface can utilize bpy library of Blender's Python API, thereby allowing manipulation of geometries, materials, camera, and render functions.

4.2.The Claude MCP Client

Linguistic abilities and planning abilities come along with Claude itself. It parses user input and recognizes the necessary functions that are to be executed in what sequence in order to reach the required objective. This is done on the basis of the

definition of each particular tool that comes along with the plugin, which provides Claude with instructions on how to perform the analysis. Claude calls tools one after another, processes the output in JSON format, and determines the next necessary action. In this way, the system manages to execute even such a complicated procedure, like observing the scene, making a floor, texturing it with wooden material, obtaining an HDRI from PolyHaven, setting it up as lighting, and importing some furniture, all that just on the basis of one request from the user.

4.3.Command Router and Handler Modules

The execute_command_internal method is responsible for executing the corresponding command handling methods based on the command received. These are the command handling methods: Code Execution, PolyHaven, Hyper3D, Sketchfab, and Scene Info. The command handling method for Code Execution helps execute the Python script necessary for using the Blender API. The command handling method for PolyHaven helps us search, download, and import assets and textures from the PolyHaven.io site. The command handling method for Hyper3D helps us create and import AI mesh jobs. The command handling method for Sketchfab helps us search, download, and import models from the Sketchfab.com site.

5. IMPLEMENTATION

5.1.Communication Protocol

In order to guarantee the connection between MCP client and the plug-in working together with Blender, the communication between them occurs in the form of exchanging messages via TCP socket by means of JSON objects. There are two main fields that appear in messages: "type," that includes the information about what command should be executed; "params," that is responsible for command parameters. The structure of the basic message looks like {"type": "execute_code", "params": {"code": "..."}}, while the structure of the required message for finding oak wood textures in PolyHaven is presented as {"type": "polyhaven_request", "params": {"action": "search", "query": "oak wood", "category": "textures"}}. Due to proper planning and preparations during the design phase, the protocol provides an opportunity for all possible commands, which can be run by using the command interface, to become outputs of the program, hence preventing the possibility of executing any invalid commands. However, since creating Python scripts for

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

Blender is not one of the features implemented in Claude, it simply executes the appropriate software communicating with the API.

5.2.PolyHaven Integration

The PolyHaven is the platform from where the user will be able to get hold of various assets such as texture, HDRI, and 3D Models, all under the CC0 License. Mentioned hereinafter is the list of the three functions that may possibly be called through the usage of the PolyHaven Plugin used by us for our project in Blender. The first of these functions is the search function, wherein users would be able to search for their required asset based on the inputs provided along with the thumbnail. The second of these functions is the downloading function, wherein users can download their asset based on the resolution. The third of these functions is the set_texture function, wherein users would be able to apply the textures to their required object through the usage of the node materials of Blender.

5.3.Hyper3D Rodin Integration

Some of the materials at PolyHaven include textures, HDRI images, and 3D models, all of which are licensed under the CC0 licensing agreement. Three tasks have been accomplished by using the PolyHaven plugin in Blender, which is being developed. First, the individual is capable of searching for the required materials based on his queries and will be provided with the thumbnails of the materials. Second, an individual can choose to download the materials he requires along with their respective resolution, and finally, there is the possibility of the set_texture command in which the individual is able to apply the specific texture to the object by creating a node material in Blender. It has been successful in providing an entirely automated process in developing a node graph, where the Principled BSDF node is generated first, followed by albedo, roughness, metallic, and normal maps.

5.4.Sketchfab Integration

Nonetheless, when one experiences uncertainty on how to go about using to create the mesh of the mage of a normal piece of furniture, an architectural design, or anything else from the physical world, the implementation of Sketchfab will be the best option because one will be able to acquire different models created by others. The Search Sketchfab Models operation will be employed to search for all models that meet the provided criteria by the user in the Sketchfab API. Thereafter, one will execute the Download Sketchfab Model operation to download the chosen model into the scene.

5.5.Scene Management and Feedback Loop

Among the key functionalities that were supposed to be incorporated are providing Claude with information on scene state. Among the key functionalities that one can use Scene Info for are get_scene_info, which gives information on all the objects in terms of their types, positions, and materials, get_object_info, which gives information on the specific object, and finally, get_viewport_screenshot, which allows Claude to take a screenshot of the scene in Base64 format. It is important for a user to possess the skill of taking a screenshot through the use of Scene Info because it assists Claude in determining if the created scene meets the requirements of the users.

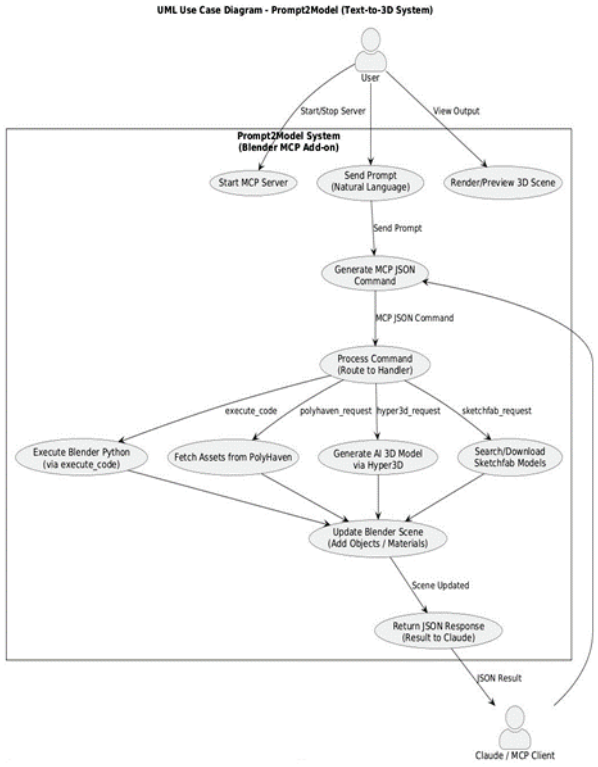


Figure 2: Use Case Diagram representing the various use cases formulated based on UML notation and their interaction between actors and objects. Actors and objects consist of user, Prompt2Model Blender Add-on, and Application client (Claude/MCP).

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

6. RESULTS AND DISCUSSION

6.1. Test Scenarios

In terms of how the model was tested, there were several prompts of varying levels of difficulty with regards to their compositions. Prompts that were not difficult at all, such as "Create a table using wood material and four legs" and "Insert a light source five meters away from the current position," were successfully completed without difficulty. On the other hand, prompts that involved more complicated compositions, like "Generate a comfortable living room containing a couch, bookshelves, wooden flooring, proper lighting, and indoor plants," will definitely take some time to complete; however, they will eventually generate the intended composition.

Figure 3: Interior scene of living room rendered using only the Prompt2Model method.

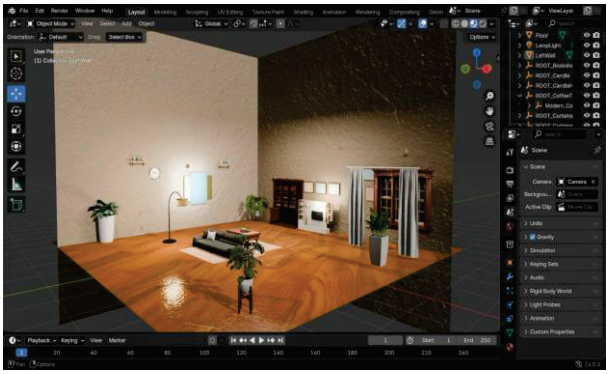
However, if we take into account that the prompts employed in the generation of the exterior scene above are taken into consideration, then it might have been possible to obtain other scenes altogether. This is illustrated further in the following figure below shown in Figure 4. The illustration below represents the results from a prompt indicating an urban street scene. Therefore, in this case, it will be important to create the facades of buildings, cars, lighting, and grunge in this particular scene.

Figure 4: Street scene of the city created using the description language.

6.2. Performance Observations

The time required to complete all the processes depended on their complexity. The time required to complete an activity easily through use of the shapes was estimated to be two minutes. Concerning the process for importing files through Sketchfab, the time required to complete the task was thirty seconds to two minutes. Concerning the process for the import file by use of Sketchfab, the time spent to complete the task was between thirty seconds to two minutes. The process using Hyper3D Rodin consumed much time by Claude than the other processes for completing the whole process. The time required to create the mesh using the AI technology was one minute to four minutes. But Claude did not have problems in giving the instructions and receiving the results from the tasks he had done with regards to the Rodin AI Mesh Generation Module. Rodin AI was able to perform other processes so long as these other processes were being performed concurrently. Errors occurred when developing a program by use of Blender. Data concerning the creation of texture using Blender is non-existent.

6.3. Limitations



Other limitations involve several more that should not go undisclosed. One of them is that the script relies on Claude's knowledge about the Python API of Blender, which is very detailed; however, his knowledge may differ slightly from the actual API, since some issues arise when writing the code through the API in Blender 4.x without any errors being reported.

7. CONCLUSION AND FUTURE WORK

However, according to Prompt2Model, the combination of all three components: a language model, a protocol of communication, and an external system integration, could help create the generation of visual 3D scenes by using natural language only in the form of a commercial 3D software. Even though it surely is not something which could take place of human creativity and eyesight (since it doesn't know that sofa should face the TV, and that there are lights on the streets), it does succeed in performing the process of creation adequately enough, so that the artist can focus more on their creative ideas, rather than on the mechanics of it.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

Below is a list of certain development approaches that we think should be discussed. The ability to introduce input data in the form of image or text interface could be regarded as an effective means of creating a connection between them. Furthermore, another possible solution for the matter under consideration lies in connecting our platform to Blender Renderer and configuring the rendering through Cycles. It will be rather difficult to address the challenge linked to latency and hallucinations because of the complexity of the process when using the API. Nevertheless, tuning the fast Claude within Blender Python script might prove helpful.

The architectural design as provided in the picture above of the MCP approach to facilitate communication between the software application and the LLM through the use of the APIs is the best architectural approach to develop such applications. Besides being rather easy to implement, it has proven to be more effective as compared to other approaches that would require LLM only to write the codes.

8. REFERENCES

[1] B. Mildenhall, P. P. Srinivasan, M. Tancik, J.
[2] T. Barron, R. Ramamoorthi, and R. Ng, "NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis," in Proc. ECCV, 2020, pp. 405-421.
[3] B. Poole, A. Jain, J. T. Barron, and B. Mildenhall, "DreamFusion: Text-to-3D Using 2D Diffusion," arXiv preprint arXiv:2209.14988, 2022
[4] 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. CVPR, 2023.
[5] R. Chen, Y. Chen, N. Jiao, and K. Jia, "Fantasia3D: Disentangling Geometry and Appearance for High-quality Text-to-3D Content Creation," in Proc. ICCV, 2023.
[6] A. Nichol, H. Jun, P. Dhariwal, P. Mishkin, and M. Chen, "Point-E: A System for Generating 3D Point Clouds from Complex Prompts," arXiv preprint arXiv:2212.08751, 2022.
[7] N. Viswanathan, S. Goldie, P. Liang, and D. Jurafsky, "Prompt2Model: Generating Deployable Models from Natural Language Instructions," arXiv preprint arXiv:2308.12261, 2023.
[8] Q.-C. Tu, T.-J. Mu, and R. R. Martin, "3D
[9] Shape Generation and Completion through Point- Voxel Diffusion," in Proc. ICCV, 2021.
[10] Anthropic, "Model Context Protocol," https:// docs.anthropic.com/en/docs/mcp, accessed March 2026.
[11] PolyHaven, "Poly Haven — The Public 3D Asset Library," https://polyhaven.com, accessed March 2026.
[12] Hyper3D, "Rodin 3D Generation API," https://hyper3d.ai, accessed March 2026.