Evaluating the Performance of Large Language Models in 3D Architectural Modeling

Aris Budhiyanto[1]*, Kevin Harsono[2], Sarai Montiel Escandon[3]

[1] Department of Architecture, Petra Christian University, Siwalankerto No.121-131, Surabaya, Indonesia
[2] Graduate Institute of Architecture, National Yang Ming Chiao Tung University, No. 1001, Daxue Rd, East District, Hsinchu City, 30010, Taiwan
[3] Department of Architecture, Universidad Veracruzana, Calz Juan Pablo II 1193, Costa Verde, 94294 Veracruz, Ver., Mexico

Keywords:
Large language model; rhino 3D model; ChatGPT; claude.

Corresponding author:
Aris Budhiyanto
Department of Architecture, Petra Christian University, Siwalankerto No.121-131, Surabaya, Indonesia
Email: arisb@petra.ac.id

Abstract

This study evaluates the performance of two large language models (LLMs), Claude and ChatGPT, in generating 3D architectural models from textual and visual prompts. Using Pass@k metrics (k = 1, 3, 5) and error analysis, the research examines their accuracy and reliability. Results show that both models perform equally at Pass@1 (24%), but Claude achieves higher scores at Pass@3 (43% vs. 35%) and Pass@5 (47% vs. 40%), demonstrating greater consistency. While syntax errors are minimal (below 5%), structural errors dominate, particularly in complex tasks where outputs tend to be simplified, incomplete, or overlapping. Logical and visual-semantic errors occur less frequently but further reduce usability. Despite these limitations, LLMs demonstrate strong potential in multimodal processing and efficiency, as they can generate models in under one minute. The findings suggest that LLM-based modeling is still at an early stage but may evolve into a transformative tool in architectural design, bridging conceptual speed with structurally valid outputs.

INTRODUCTION

In recent years, the use of artificial intelligence (AI) has been growing rapidly, especially following the emergence of ChatGPT in late 2022. This has led to the implementation of various software tools designed to support, or even replace, human work (Sugiarto, 2025). In architectural practices, AI plays a vital role across multiple aspects, from conceptual design and material or systems analysis to construction workflows, building management, and the documentation of built environments. Its application involves every stage of the design process: starting with early-stage brainstorming, continuing through design optimization and environmental simulation, to generating technical drawings, including floor plan, facade, structural system, and section, and supporting visualization, rendering, and final presentation phases (Avinç et al., 2025; Li et al., 2025; Redyantanu et al., 2024; Redyantanu & Dharmatanna, 2025). Historically, the integration of AI into architecture has been started since the 1960s,

when computational tools were first employed to assist in design processes. This was followed in the 2000s by the emergence of parametric design, supported by tools such as Revit, Rhino3D, and Grasshopper, which employ algorithms and variables to produce dynamic and adaptive forms.

Recently, AI and machine learning (ML) have been used to assist designers in creating more complex and efficient designs (Avinç et al., 2025; Hegazy & Saleh, 2023). ML generative tools, such as text-to-text, image-to-image, and text-to-image, have been integrated into the architectural design process (Horvath & Pouliou, 2024; Shema & Abdulmalik, 2024). Some notable research in this area includes holistic optimization using parametric scripting and building performance simulation to analyze the building performance during the early stage (Nembrini et al., 2014). Exploration of generative tools for site analysis and design development using text-based prompts, including architectural concepts, visualization, and 3D models, was conducted by Agkathidis et al. (2024). Another study examines how text-to-image generative tools can support the development of architectural design ideas (Albaghajati et al., 2023).

Rhino 3D, an NURBS-based 3D model, is widely used not only in the architectural domain, to construct parametric models. (Mohol & Bambawale, 2023; Shtepani & Yunitsyna, 2024). One advantage of using this software is that the user can create their own application programming interface (API), enabling them to create their custom scripts, plug-ins, and standalone applications that support the automation workflows, parametric modelling, and integrating Rhino 3D with other digital tools (Talkhounche, 2024). Moreover, Rhino 3D provides scripting-based modelling, which paves the way to integrate large language models (LLMs) and the architectural design process (Chairiyah et al., 2022). The application of this integration workflow, presented by Horvath & Pouliou (2024), explores their design method using text-to-text, text-to-image, and image-to-image ML tools to construct a parameterized skyscraper tower.. Basarir & Erol (2021) examined the use of Generative Adversarial Networks (GANs) to translate architectural briefs into visual outputs, while Yang et al. (2023) introduced an ML framework that changes hand-drawn 2D sketches into 3D models. Dai et al. (2023) studied an AI–human collaborative design framework that supports users to explore, generate, and improve conceptual forms through a combination of textual prompts, 2D images, and 3D models in Rhino/Grasshopper. Natividad-Vivó et al. (2022) reviewed various Rhino procedures for drawing ellipses and introduced a general method using Python Script to make the ellipses from any circumscribed convex quadrilateral. Kang (2024) used generative AI to generate code using ChatGPT prompting and to create 3D models in Rhino using Rhino Python Script.

The integration of LLMs into Rhino modeling can be considered as a form of resilience in architecture, as LLM functions as a dynamic control layer supporting the system's ability to adapt, recover, and evolve. Language as a medium of interaction lets architects to quickly respond to design changes, monitor performance in real time, anticipate environmental and functional challenges, and learn from previous results. This supports resilience in both the architectural product and the design process, ensuring robustness and adaptability when facing uncertainty and complexity design (Tomassi et al., 2025). However, further exploration is still needed, particularly related to utilizing various advanced LLMs and developing more complex designs (Hizmi et al., 2025). The objective of this study is to explore and analyze the application of LLM in AI-based parametric architectural modelling using Rhino, while mapping its potential and evaluating the performance of the generated designs.

LITERATURE REVIEW

The LLM is a type of neural network model, specifically based on the transformer architecture, designed to understand and generate human language (Chang et al., 2024). It was trained on a very large text dataset that was made up of many parameters, which allows it to solve complex language tasks, including translation, summarization, code generation, and dialogue. One of LLM's potentialities is its ability for in-context learning, where the model can adjust to different tasks directly from a prompt without extra fine-tuning (Kasneci et al., 2023; Li et al., 2025; Talkhounche, 2024). Two well-known transformer-based approaches are BERT (Bidirectional Encoder Representations from Transformers) and GPT (Generative Pretrained Transformer) (Li et al., 2025). ChatGPT from OpenAI and Claude 2 from Anthropic are among the most widely used GPT-based chatbots, and they are recognized for producing coherent and human-like responses (Borji & Mohammadian, 2023; Rivas & Zhao, 2023; Wu et al., 2024).

In architecture, LLMs have started to be used for generating both 2D images and 3D models, through text-to-image, image-to-image, and text-to-3D applications (Fig 1) (Li et al., 2025; Tan & Luhrs, 2024). Text-to-image generators create images based on written descriptions and rely on large image datasets with text labels—such as ImageNet, which contains over 14 million pictures labeled using nouns from a database called WordNet. The idea is that nouns represent objects that can be depicted in pictures, helping AI learn to recognize them. Image-to-image generators, an AI tool based on GANs, use existing images as input to create new, related images. Several studies have used GANs to automatically generate floor plans, building facades, perspective views, and even architectural concepts from simple sketches. Similar to text-to-image, text-to-3D generators use natural language descriptions to create detailed 3D models without relying on large 3D datasets. While in image-to-3D generation, a 3D model can be reconstructed from just one or a few images (Agkathidis et al., 2024; Horvath & Pouliou, 2024; Li et al., 2025; Rodriguez et al., 2025).

Fig. 1. Examples of Generated Results from Text-to-image (A), Image-to-image (B) and Image-to-3D (C). (Source: Agkathidis, 2024; Li et al., 2025)

METHODS

Research Design and Tools

This study adopts an experimental design to evaluate the capability of LLMs in generating architectural 3D modeling scripts for Rhino 3D. While Rhino 3D does not natively support a chatbot interface, the recent development in AI LLM has opened new ways to assist with 3D modeling, scripting, and design exploration. ChatGPT-4 developed by OpenAI, and Claude Sonnet 4 by Anthropic are used in this study as the main LLMs for generating architectural 3D modelling scripts. These two LLMs were selected due to their superior performance compared

to other open-source LLMs (Wang et al., 2024; Wu et al., 2024). While ChatGPT-4 can turn natural language prompts into code, it cannot directly create 3D models. Therefore, Python scripts produced by ChatGPT are executed within the Rhino 3D environment through its scripting interface (Fig. 2) (Kang, 2024). Similarly, Claude is connected to Rhino 3D using the RhinoAiMCP plugin (Fig. 3). This plugin includes a built-in Model–Context–Protocol (MCP) server that enables Rhino 3D and the AI model to communicate in real-time (RhinoAI-MCP, 2024).

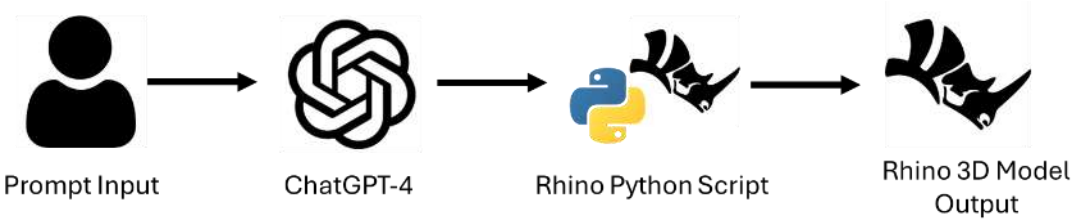


Fig. 2. Integrated Workflow Framework of ChatGPT and Rhino 3D via Python Scripting. (Source: author)

Fig. 3. Integrated Workflow Framework of Claude and Rhino 3D using RhinoAiMCP plugin. (Source: author)

Prompt Design

Both LLMs were given the same set of prompt inputs, consisting of texts and images with varying levels of complexity, ranging from simple to highly complex. The text prompts were as follows:

1. Create a simple building with dimensions of 10×10×3 m^3, with a single door measuring 1×2 m^2 located at the center of the north side.
2. Model a cylinder with a diameter of 3 meters and a height of 2 meters, positioned horizontally, with the lateral surface colored yellow and the circular faces colored red.
3. Create a composition of four identical cubes, each measuring 3×3×3 meters, arranged in a single row with a 5-meter gap between each cube.
4. Create a 3-story building model measuring 10×10 meters with a total height of 10 meters, featuring five windows (1×1 meter each) on every side of the façade.
5. Create an open cylindrical pavilion with a 10-meter diameter, a flat roof, and ten evenly spaced columns (each 1 meter in diameter) arranged in a circle around its perimeter.
6. Create a stilt house model measuring 3×4 meters with a total height of 3 meters, featuring a gable roof, an elevated floor 1 meter above the ground, and supported by columns at all four corners.
7. Model a two-story building measuring 5×5 meters, with each floor 3 meters high and a hip roof. The west façade includes external eggcrate shading with a grid size of 0.5×0.5 meters per opening. The north façade uses external vertical louver shading, with 1-meter spacing between louvers and a louver length of 0.5 meters.

8. Create a model of the Guggenheim Bilbao Museum.
9. Model a 10-story building with no walls on the north side, a floor plan of 5×5 meters, and a floor-to-floor height of 3 meters. The building features a vertical central void (atrium) measuring 2×2 meters, surrounded by a spiral staircase, and a skylight at the top measuring 2×2 meters.

Among these text prompts, Prompts 1–3 are categorized as simple, Prompts 4–6 as moderately complex, and Prompts 7–9 as complex. While Fig. 4. Presents image prompts with varying levels of complexity: Images A-B are categorized as simple, Images C-D as moderately complex, and Images E-F as complex, detailed drawing.

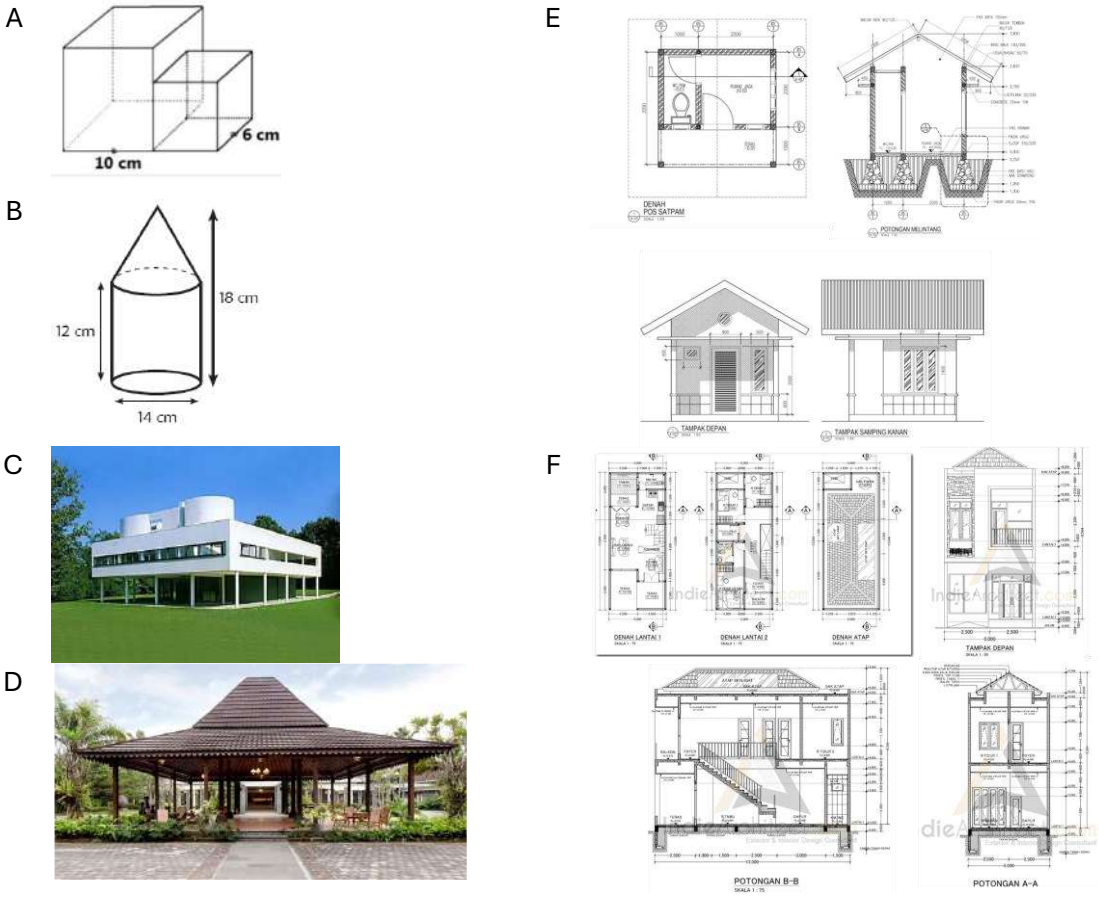


Fig. 4. Figure Prompts with Different Complexity.
(Source: modified by author)

Experimental Procedure

Code generation systems are intended to improve development efficiency by producing a correct solution within a small set of candidates, which is commonly assessed using pass@k (usually k = 1, 5, or 10); if users must test more than ten outputs, the efficiency benefits are lost, yet ensuring a correct result within this limited range remains a persistent challenge for current LLM-based approaches (Lyu et al., 2025). The pass@k metric, widely adopted in code generation research (Du et al., 2023; Kulal et al., 2019), measures the probability that at least one correct solution exists among k independent attempts, thereby capturing how consistently a model can produce valid outputs across multiple trials.

The experimental procedure was carried out as follows:

a. Input to LLMs: Each text or image prompt was submitted to both ChatGPT-4 and Claude Sonnet 4 under default API settings. No explicit decoding parameters were altered, thereby reflecting default robustness.
b. Re-runs (Manual Replication): Unlike previous studies that rely on stochastic decoding techniques such as nucleus sampling (top-p) or temperature-based sampling to generate diverse outputs (Chen et al., 2025; Kulal et al., 2019), this study adopts a manual multi-trial prompting strategy. Each prompt was executed five times. This produced multiple Python script outputs under identical conditions.
c. Execution in Rhino 3D: Each script was run in the Rhino 3D environment. A result was counted as successful if the script (a) executed without syntax or runtime errors and (b) produced a model conforming to the requirements of the original prompt.
d. Error Logging: Scripts that failed to run or produced incorrect geometry were logged as unsuccessful outputs.

Evaluation: For each set of five outputs, the results were classified as correct or incorrect based on their structural validity, logical consistency, and compliance with the intended design. Based on this classification, pass@k metric is calculated for k = 1, 3, and 5 to evaluate the success rate across multiple attempts. Pass@1 represents the accuracy when only the first attempt is considered. Pass@3 measures the probability of success when three independent attempts are allowed. Pass@5 reflects the success rate when all five attempts are considered.

$$pass@k = E_{problems}\left[1 - \frac{(n-ck)}{(nk)}\right] \quad \text{..... (1)}$$

Where:

$E_{problems}$ = average taken across all problems
n = the total number of generated samples per problem.
c = the number of correct samples.
k = the number of samples allowed to check

Additionally, the types of errors encountered during the experiments are observed. To gain clearer insights, the observed mistakes are classified into the following categories (Yuan et al., 2024):

1. Syntax Errors: These involve problems in the modeling script or code that prevent the model from running or compiling properly.
2. Geometric Errors: These are further divided into two subcategories:
   – Structural Configuration Errors: Occur when the spatial arrangement of elements significantly contradicts the original design prompt or description.
   – Spatial Precision Errors: Involve small inaccuracies in the geometry, such as slight misalignment, incorrect scaling, or improper rotation of components.
3. Logical Errors: These stem from a failure to follow basic real-world logic or common-sense reasoning, leading to unrealistic or functionally flawed outcomes.

RESULTS AND DISCUSSION

Both LLMs can recognize and process the given prompts, as they correctly identify the images of Villa Savoye, the Joglo pavilion, and the guard post in Fig. 4. They also recognize the Guggenheim Bilbao Museum as Frank Gehry's masterpiece, noted for its distinctive curved titanium forms and complex geometry. However, Claude shows an advantage over ChatGPT, as it can automatically detect errors and generate a revised script to address them. The 3D model outputs are presented in the Appendix.

The experiment shows that Claude and ChatGPT perform the same at Pass@1, with both models reaching 24%. This means that if only one answer is generated, their accuracy is equal. However, the difference becomes clear when more attempts are allowed. At Pass@3, Claude scores 43% compared to ChatGPT's 35%, showing that Claude is more likely to produce at least one correct answer within three tries. The gap increases further at Pass@5, where Claude reaches 47% while ChatGPT scores 40% (Fig. 5). These results suggest that Claude is more consistent and reliable across multiple attempts. At the same time, ChatGPT tends to perform slightly weaker when more outputs are considered. This is because Claude can automatically detect and correct the errors in the script generated.

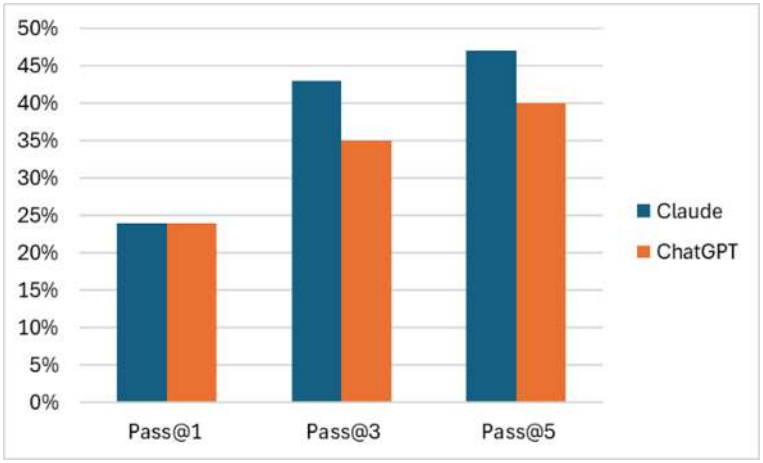


Fig. 5. Experiment Result Pass@k for Claude dan ChatGPT.
(Source: author)

The low scores indicate that more failure models were generated than correct ones. Figure 8 shows that Claude and ChatGPT generate correct models in only approximately 21% and 23% of cases, respectively. Although both LLMs almost always succeed in generating 3D models and rarely make code-level mistakes, as indicated by the number of syntax errors occurring in less than 5%, the majority of outputs are classified as structural errors, as this error dominates more than half of the model outputs in both LLMs. This indicates the main weakness of the LLM method in generating 3D models lies not in syntax but in generating structurally valid designs that align with the intended outcomes (Fig. 6).

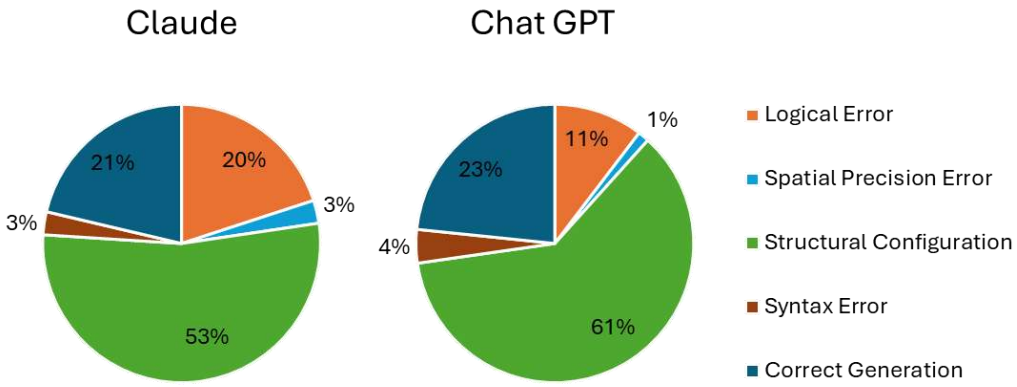


Fig. 6. Statistical Analysis on Error Cases.
(Source: author)

Fig. 7 shows several cases of these structural configuration errors, where parts of the structure overlap, remain incomplete, or are oversimplified. For example, Fig. 9A shows two boxes intersecting each other, while Fig. 9B and Fig. 9C display incomplete structural components. Fig. 9D illustrates an oversimplified model. In most cases, the errors come from incomplete or simplified outputs, especially in models with higher complexity. This shows that the LLMs tend to reduce details and simplify the structure when facing more difficult tasks, which explains why structural errors dominate the experimental results.

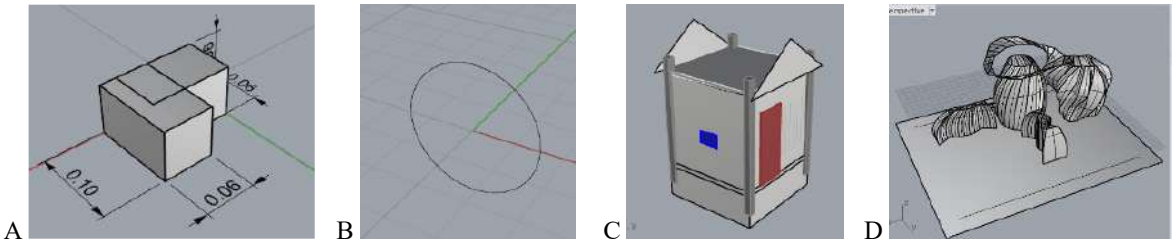


Fig. 7. Structural Configuration Error.
(Source: author)

Logical errors are the second most frequent type of error, although their occurrence is still lower than the number of correct models. The proportion of logical errors reaches 20% in Claude and 11% in ChatGPT, showing that both LLMs occasionally generate outputs that are structurally complete but logically inconsistent (Fig. 8). This error does not prevent the model from being generated, but it reduces the usability of the output since the geometry or placement often does not follow the expected design logic. Fig. 8 presents several examples of logical errors. In Fig. 8A, the door is generated not on the wall surface but instead crossing through it. In Fig. 8B, the cone is inverted, creating an unrealistic geometry. In Fig. 8C, the building layout shows that the shading devices are detached from the façade.

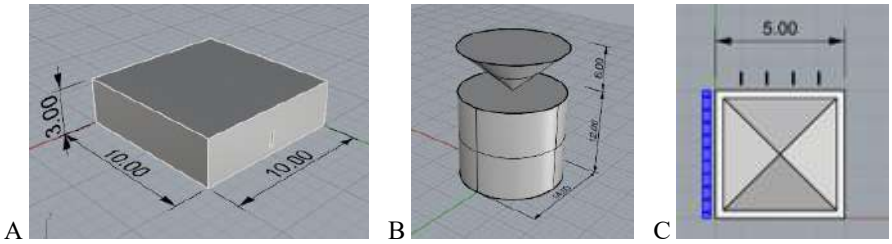


Fig. 8. Logical Error
(Source: author)

Another type of logical error is the visual-semantic error (Zhang et al., 2024). As shown in Fig. 9B and 9C, the cylinders fail to display the specified color, even though their dimensions and sizes are correct. Another limitation observed in both LLMs is the inconsistency of color rendering. For example, in Fig. 9A and 9B, the color appears only when the display mode is set to 'Rendered', while in Fig. 9C and 9D, the color is visible only in 'Shaded' mode. Since the prompt does not specify a particular display setting, the output is considered acceptable if the correct color appears in at least one display mode.

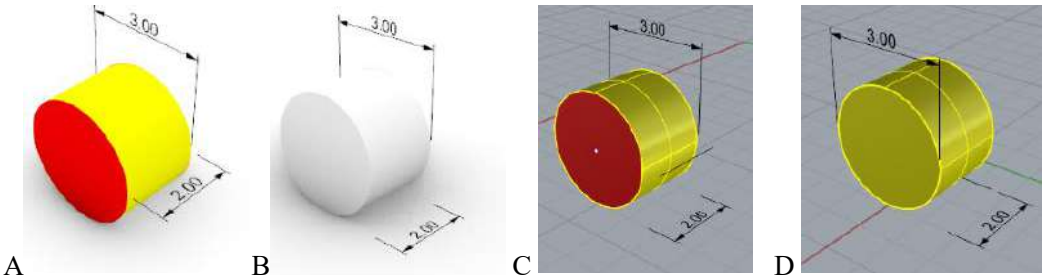


Fig. 9. Visual-Semantic Error.
(Source: author)

Considering that both LLMs demonstrate very low performance at Pass@1 and only a slight increase at Pass@3 and Pass@5, this indicates that correct results are rarely generated across a wide range of samples. When confronted with complex prompts or images, the models tend to produce outputs with structural errors, leading to the tendency of both Claude and ChatGPT to simplify objects and their limitations in generating complex models (Chang et al., 2024). While the LLM used to generate simple and well-defined objects, the error rate is considerably lower. On the other hand, the method has proven effective for producing familiar and simple objects; its accuracy decreases when generating outputs for complex prompts (Hizmi et al., 2025). This indicates that the consistency of LLMs in generating precise models is very low (Albaghajati et al., 2023). Compared to Claude, which can detect and correct errors automatically, ChatGPT still needs feedback from the user to resolve syntax errors and produce the intended design (Kang, 2024).

Although the current LLM application for generating 3D models remains limited, its potential is still considerable. These models can process multimodal inputs, both text and images, and translate them into a real three-dimensional form (Wu et al., 2024). Moreover, it only took a few minutes to create the models, which is a great advantage to save time for modelling and reduce the effort required for manual modelling (Albaghajati et al., 2023). Based on the findings, this method is still in an early developmental stage and needs some improvements to address the limitations. In the future, LLMs and generative AI have the

potential to develop into powerful tools that affect and transform the architectural design process (Agkathidis et al., 2024; Mohol & Bambawale, 2023).

Regarding the resilience in architecture, this study has several implications contributing to the architectural design process. First, this study contributes to the utilization of AI in architecture by showing that natural language and multimodal prompts can be integrated into the design process. This method connects human conceptual thinking with computational modelling, leading to improvement in design modelling practices. The architects may describe ideas using natural language instead of manually modelling their ideas using 3D modelling software (Shin et al., 2025). Second, in architecture education, LLMs can be used as rapid prototyping assistants, which allow students to experiment with form, geometry, and structure quickly. This helps support students' exploration and creative learning (Kampelopoulos et al., 2025). Although this method has low-accuracy outputs, which make it not yet reliable for professional practice, it shows AI potential in architectural work, which can reduce repetitive tasks, generate early-stage design alternatives, and help with feasibility studies (Jin et al., 2024). Third, this study introduces a framework for evaluating AI-based generative tools in design that can be future adopted or improved in the future (Onatayo et al., 2024).

CONCLUSION

This study shows the potential and the limitations of using LLMs to generate 3D models from text and image prompts. Both Claude and ChatGPT recognize architectural objects. This recognition indicates their ability to perform multimodal recognition. Claude outperforms ChatGPT because it can automatically detect and correct errors without human aid. This is evident in the higher Pass@3 and Pass@5 scores.

Although both LLMs can generate 3D models, the performance remains low. Most failures are dominated by structural errors indicated by incomplete forms, intersecting geometries, or oversimplified structures, particularly when the input prompts are complex. Furthermore, models are also limited in their ability to perform logical and visual-semantic reasoning, thereby lacking the capacity to generate designs that are rational and cohesively aligned visually as well as structurally. This emphasizes the key limitation of the LLM-based modeling, which tends to overly simplify objects and lack rational control over the complexity.

Despite the limitations, this method has great potentials. LLMs can process multimodal information by combining text and image inputs. They also deliver models less than a minute, which is a significant time efficiency compared to manual modelling. In addition to this efficiency, the rapid, significant improvement of LLMs make them as a potential tool for architectural workflows. However, this method still needs a lot of substantial improvement before they can consistently produce outputs with the precision and reliability expected in professional design practice.

In architectural design process, this study provides several contributions. First, it introduces the discussion about AI in architecture by showing how multimodal prompts and natural language can be used to create design models. This method connects human thinking with computational modeling. Second, it leads a way where architects and students can use these systems to quickly make prototypes and try out different ideas. Third, it presents an evaluation methodology that can be used to assess AI-driven architectural tools in the future. Furthermore,

this study shows not only the current condition of technology but also its potential impact on architectural design practice and education in the future.

REFERENCES

Agkathidis, A., Song, Y., & Symeonidou, I. (2024). AI-Assisted Design Utilising artificial intelligence as a generative form-finding tool in architectural design studio teaching. *Data-Driven Intelligence: Proceedings of the 42nd Conference on Education and Research in Computer Aided Architectural Design in Europe*, 619–628.

Albaghajati, Z. M., Bettaieb, D. M., & Malek, R. B. (2023). Exploring text-to-image application in architectural design: insights and implications. *Architecture, Structures and Construction*, *3*(4), 475–497. https://doi.org/10.1007/s44150-023-00103-x

Avinç, G. M., Aycı, H., & Taş, A. (2025). The Role of Artificial Intelligence in Architecture: The Case Studies of Star Architects. *| Dynamic Impact Architects. Rupkatha Journal*, *17*(1), 17. https://doi.org/10.21659/rupkatha.v17n1.04

Basarir, L., & Erol, K. (2021). Briefing AI: From Architectural Design Brief Texts to Architectural Design Sketches. *9th International Conference of the Arab Society for Computer Aided Architectural Design*, 23–31. https://www.researchgate.net/publication/349734427

Borji, A., & Mohammadian, M. (2023). Battle of the Wordsmiths: Comparing ChatGPT, GPT-4, Claude, and Bard. *GPT-4, Claude, and Bard*. https://doi.org/dx.doi.org/10.2139/ssrn.4476855

Chairiyah, R., Yetti, A. E., & Pujiyanti, I. (2022). *The Grasshopper+Rhino for 3D Modelling in Indonesian's Education of Biomimetic Architecture*. www.rhino3d.com

Chang, Y., Wang, X., Wang, J., Wu, Y., Yang, L., Zhu, K., Chen, H., Yi, X., Wang, C., Wang, Y., Ye, W., Zhang, Y., Chang, Y., Yu, P. S., Yang, Q., & Xie, X. (2024). A Survey on Evaluation of Large Language Models. *ACM Transactions on Intelligent Systems and Technology*, *15*(3). https://doi.org/10.1145/3641289

Chen, B., Zhang, Z., Langrené, N., & Zhu, S. (2025). Unleashing the potential of prompt engineering for large language models. *Patterns*, *6*(6). https://doi.org/10.1016/j.patter.2025.101260

Dai, S., Li, Y., Grace, K., & Globa, A. (2023). Towards Human-AI Collaborative Architectural Concept Design via Semantic AI. In M. Turrin, C. Andriotis, & A. Rafiee (Eds.), *Computer-Aided Architectural Design. INTERCONNECTIONS: Co-computing Beyond Boundaries. CAAD Futures 2023.* Communications in Computer and Information Science.

Du, X., Liu, M., Wang, K., Wang, H., Liu, J., Chen, Y., Feng, J., Sha, C., Peng, X., & Lou, Y. (2023). ClassEval: A Manually-Crafted Benchmark for Evaluating LLMs on Class-level Code Generation. *ArXiv Preprint ArXiv:2308.01861*. http://arxiv.org/abs/2308.01861

Hegazy, M., & Saleh, A. (2023). Evolution of AI role in architectural design: between parametric exploration and machine hallucination. *MSA Engineering Journal*, *2*(2), 262–288. https://doi.org/10.21608/msaeng.2023.291873

Hizmi, B. El, Sterman, Y., & Austern, G. (2025). LLMto3D- Generation of parametric, 3D printable objects using large language models. *International Journal of Architectural Computing*, 1–9. https://doi.org/10.1177/14780771251353792

Horvath, A. S., & Pouliou, P. (2024). AI for conceptual architecture: Reflections on designing with text-to-text, text-to-image, and image-to-image generators. *Frontiers of Architectural Research*, *13*(3), 593–612. https://doi.org/10.1016/j.foar.2024.02.006

Jin, S., Tu, H., Li, J., Fang, Y., Qu, Z., Xu, F., Liu, K., & Lin, Y. (2024). Enhancing Architectural Education through Artificial Intelligence: A Case Study of an AI-Assisted Architectural Programming and Design Course. *Buildings*, *14*(6). https://doi.org/10.3390/buildings14061613

Kampelopoulos, D., Tsanousa, A., Vrochidis, S., & Kompatsiaris, I. (2025). A review of LLMs and their applications in the architecture, engineering and construction industry. *Artificial Intelligence Review*, *58*(8), 1–46. https://doi.org/10.1007/s10462-025-11241-7

Kang, H.-R. (2024). A Study of Jewelry 3D Modeling Using Rhino Python and Generative AI. *The Journal of the Convergence on Culture Technology (JCCT)*, *10*(6), 821–827. https://doi.org/10.17703/JCCT.2024.10.6.821

Kasneci, E., Sessler, K., Küchemann, S., Bannert, M., Dementieva, D., Fischer, F., Gasser, U., Groh, G., Günnemann, S., Hüllermeier, E., Krusche, S., Kutyniok, G., Michaeli, T., Nerdel, C., Pfeffer, J., Poquet, O., Sailer, M., Schmidt, A., Seidel, T., … Kasneci, G. (2023). ChatGPT for good? On opportunities and challenges of large language models for education. *Learning and Individual Differences*, *103*, 1–9. https://doi.org/10.1016/j.lindif.2023.102274

Kulal, S., Pasupat, P., Chandra, K., Lee, M., Padon, O., Aiken, A., & Liang, P. (2019). SPoC: Search-based Pseudocode to Code. *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, 11906–11917. http://arxiv.org/abs/1906.04908

Li, C., Zhang, T., Du, X., Zhang, Y., & Xie, H. (2025). Generative AI models for different steps in architectural design: A literature review. In *Frontiers of Architectural Research* (Vol. 14, Issue 3, pp. 759–783). KeAi Communications Co. https://doi.org/10.1016/j.foar.2024.10.001

Lyu, Z.-C., Li, X.-Y., Xie, Z., & Li, M. (2025). Top Pass: improve code generation by pass@k-maximized code ranking. *Frontiers of Computer Science*, *19*, 198341. https://doi.org/https://doi.org/10.1007/s11704-024-40415-9

Mohol, N., & Bambawale, A. (2023). *Understanding the Role of Rhino as a Parametric Tool in Designing Architectural Built-Forms* (pp. 167–187). https://doi.org/10.2991/978-94-6463-136-4_18

Natividad-Vivó, P., García-Baño, R., Salcedo-Galera, M., & Calvo-López, J. (2022). Representation of Oblique Circles in CAD Implemented in a Python Script for Rhinoceros. In *Springer Series in Design and Innovation* (Vol. 21, pp. 294–303). Springer Nature. https://doi.org/10.1007/978-3-031-04632-2_31

Nembrini, J., Samberger, S., & Labelle, G. (2014). Parametric scripting for early design performance simulation. *Energy and Buildings*, *68*(PART C), 786–798. https://doi.org/10.1016/j.enbuild.2013.09.044

Onatayo, D., Onososen, A., Oyediran, A. O., Oyediran, H., Arowoiya, V., & Onatayo, E. (2024). Generative AI Applications in Architecture, Engineering, and Construction: Trends, Implications for Practice, Education & Imperatives for Upskilling—A Review. In *Architecture* (Vol. 4, Issue 4, pp. 877–902). Multidisciplinary Digital Publishing Institute (MDPI). https://doi.org/10.3390/architecture4040046

Redyantanu, B. P., & Dharmatanna, S. W. (2025). Dari Tekstual ke Visual: Memikirkan Kembali Proses Perancangan Arsitektur. In L. T. Tjiek, I. Sugiarto, & E. A. Iskandar (Eds.), *Towards an AI-Native Campus: UK Petra x Kecerdasan Artifisial* (pp. 40–52).

Redyantanu, B. P., Yatmo, Y. A., Atmodiwirjo, P., & Sinuraibhan, S. (2024). Actual material virtual materiality: Multiplicity of locality-based exhibition. *A/Z ITU Journal of the Faculty of Architecture*, *21*(2), 411–430. https://doi.org/10.58278/0.2024.53

RhinoAI-MCP. (2024). *RhinoAI-MCP: Model-Context-Protocol for Rhino 3D*. Food4Rhino. https://www.food4rhino.com/en/app/rhinoaimcp?lang=en

Rivas, P., & Zhao, L. (2023). Marketing with ChatGPT: Navigating the Ethical Terrain of GPT-Based Chatbot Technology. In *AI (Switzerland)* (Vol. 4, Issue 2, pp. 375–384). Multidisciplinary Digital Publishing Institute (MDPI). https://doi.org/10.3390/ai4020019

Rodriguez, J. D. S., Joyce, S. C., & Julfendi, J. (2025). Using customized GPT to develop prompting proficiency in architectural AI-generated images. *Hal-05037370*.

Shema, A., & Abdulmalik, H. (2024). Artificial intelligence (AI) in architecture and design. In H. R. Husain (Ed.), *AI-driven architecture: Pioneering the digital frontier* (pp. 145–186). Alanya University. https://doi.org/10.38027/ai-driven-7

Shin, D., Chen, T.-Y., Hsieh, G., & Wang, L. L. (2025). *What About My Design Context?: Exploring the Use of Generative AI to Support Customization of Translational Research Artifacts*. 1210–1227. https://doi.org/10.1145/3715336.3735686

Shtepani, E., & Yunitsyna, A. (2024). Learning the Generative Design Tools for Architecture Projects: An Experience from a Student-Design Workshop. *3rd International Conference on Frontiers in Academic Research*.

Sugiarto, I. (2025). AI: Teritorial yang Belum Terpetakan. In L. T. Tjiek, I. Sugiarto, & E. A. Iskandar (Eds.), *Towards an AI-Native Campus: UK Petra x Kecerdasan Artifisial* (pp. 2–12). Universitas Kristen Petra.

Talkhounche, S. N. (2024). *A NATURAL LANGUAGE INTERFACE FOR MODELING IN RHINO 3D USING LARGE LANGUAGE MODELS*. The University of North Carolina.

Tan, L., & Luhrs, M. (2024). Using Generative AI Midjourney to enhance divergent and convergent thinking in an architect's creative design process. *Design Journal*, *27*(4), 677–699. https://doi.org/10.1080/14606925.2024.2353479

Tomassi, A., Falegnami, A., & Romano, E. (2025). Talking Resilience: Embedded Natural Language Cyber-Organizations by Design. *Systems*, *13*(4), 247. https://doi.org/10.3390/systems13040247

Wang, Y., Liang, L., Li, R., Wang, Y., & Hao, C. (2024). Comparison of the Performance of ChatGPT, Claude and Bard in Support of Myopia Prevention and Control. *Journal of Multidisciplinary Healthcare*, *17*, 3917–3929. https://doi.org/10.2147/JMDH.S473680

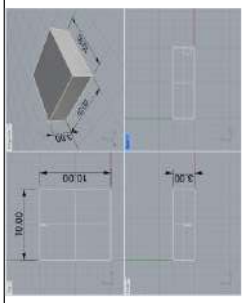
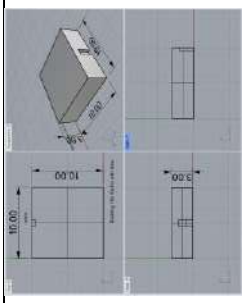
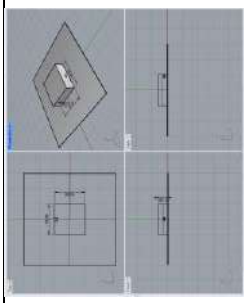
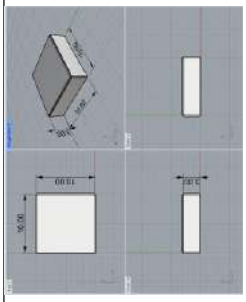
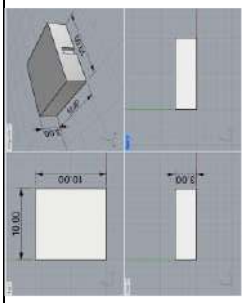
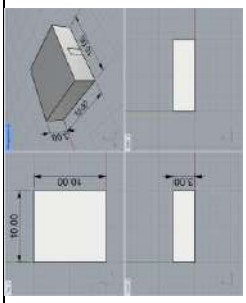
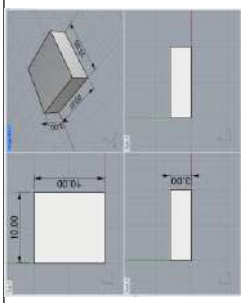
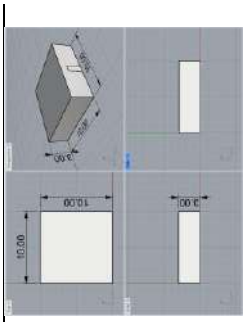
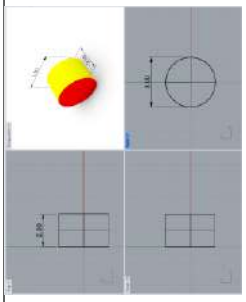
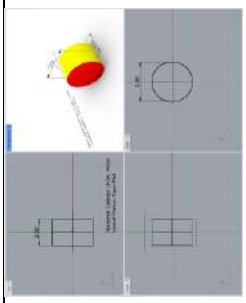
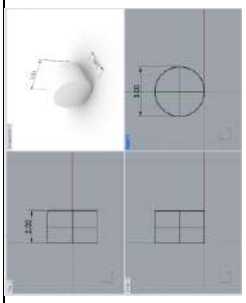
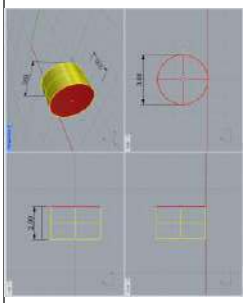
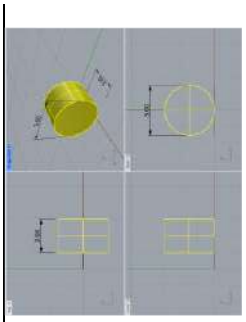
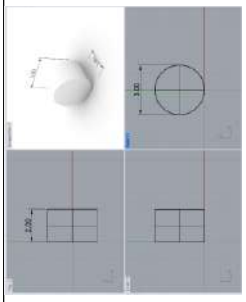
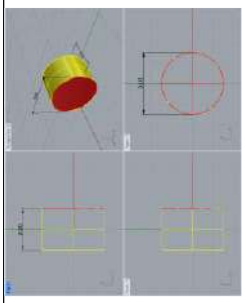
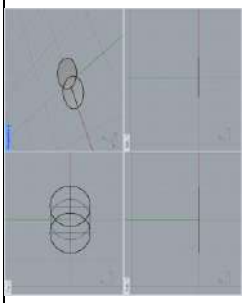
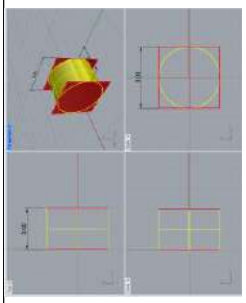
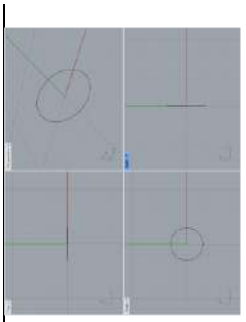
Wu, S., Koo, M., Blum, L., Black, A., Kao, L., Fei, Z., Scalzo, F., & Kurtz, I. (2024). Benchmarking Open-Source Large Language Models, GPT-4 and Claude 2 on Multiple-Choice Questions in Nephrology. *NEJM AI*, *1*(2). https://doi.org/10.1056/aidbp2300092

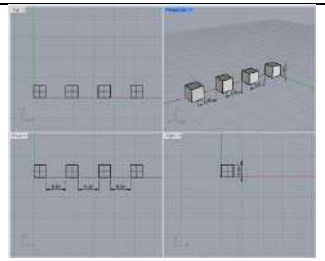
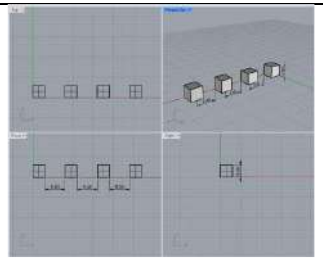
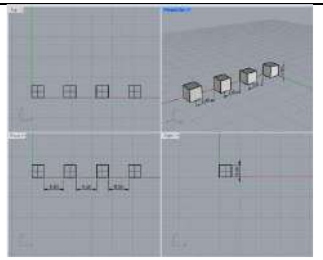
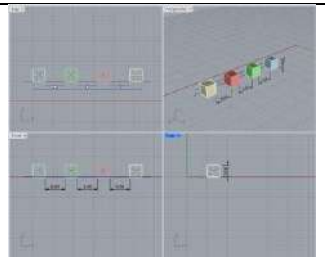
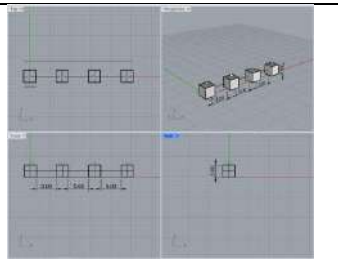
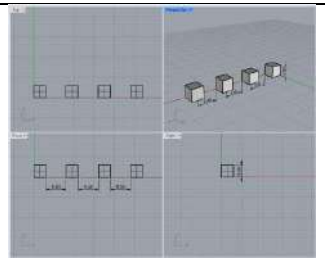
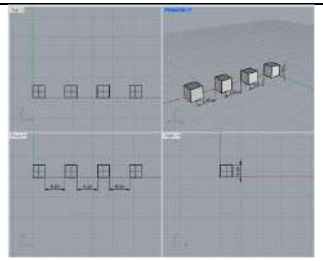
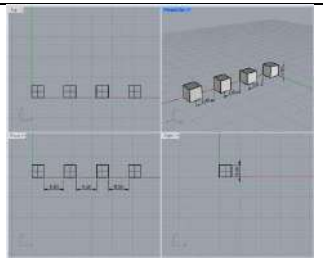
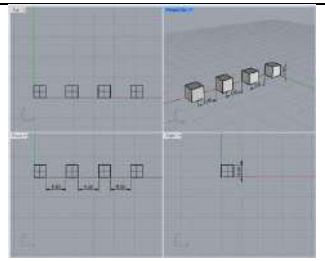
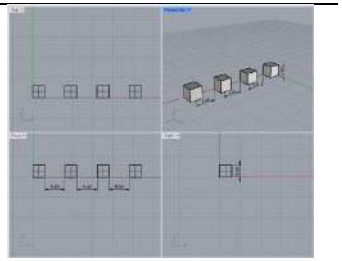
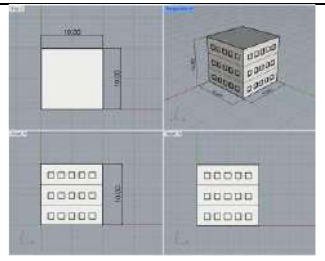
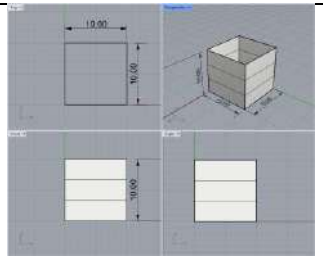
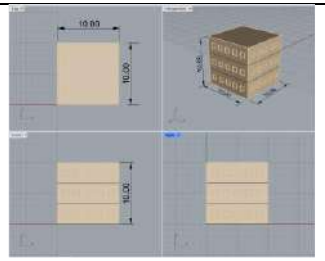
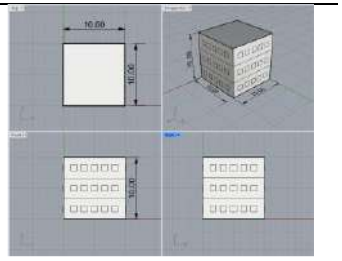
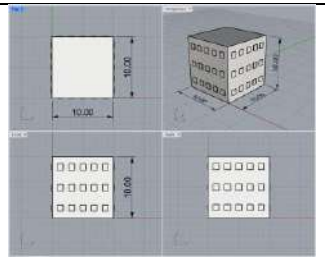
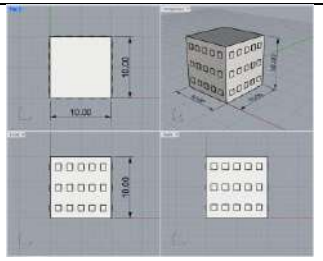
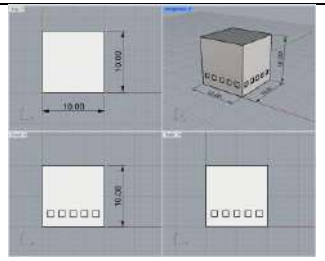
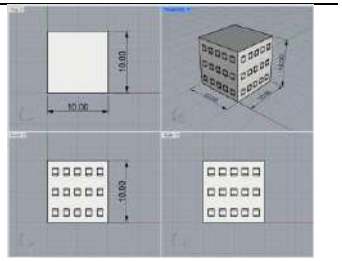
Yang, H. Bin, Johanes, M., Kim, F. C., Bernhard, M., & Huang, J. (2023). Architectural Sketch to 3D Model: An Experiment on Simple-Form Houses. *Communications in Computer and Information Science*, *1819 CCIS*, 53–67. https://doi.org/10.1007/978-3-031-37189-9_4

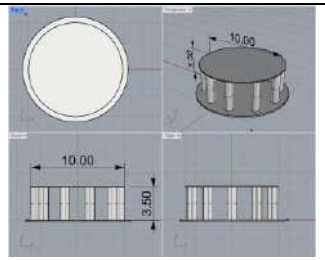
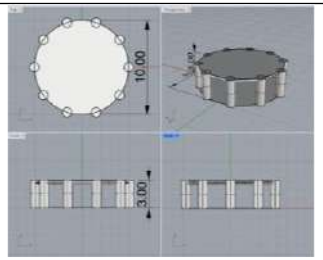
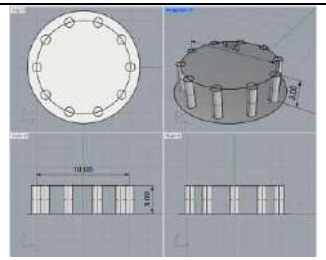
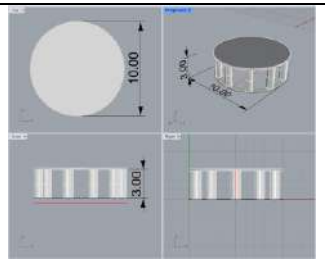
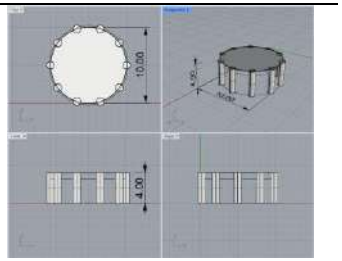
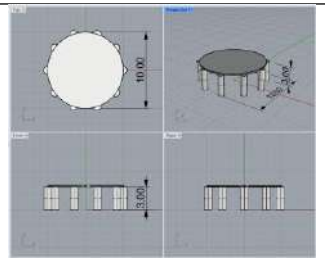
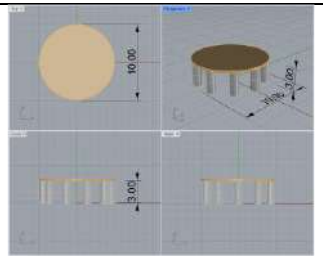
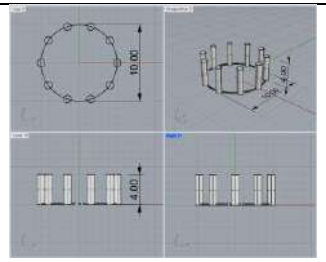
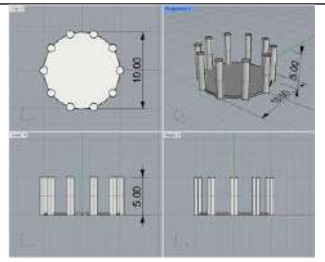
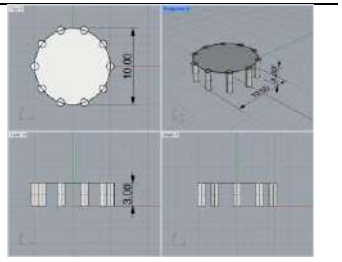
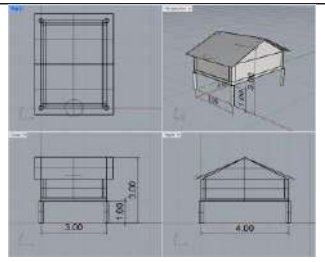
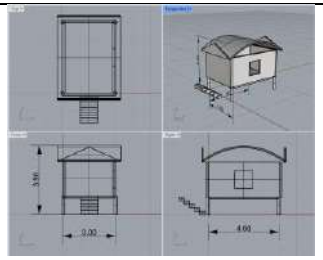
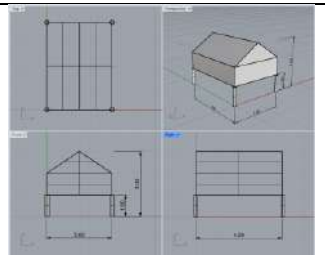
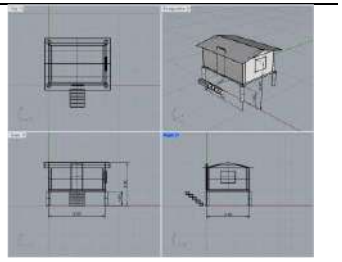
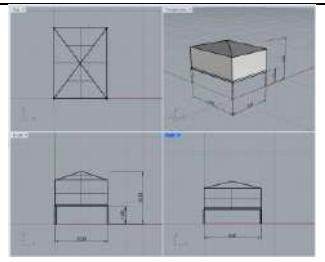
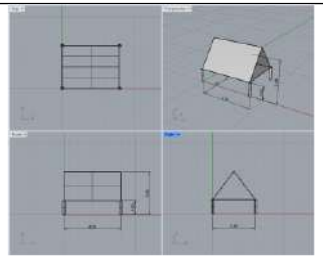
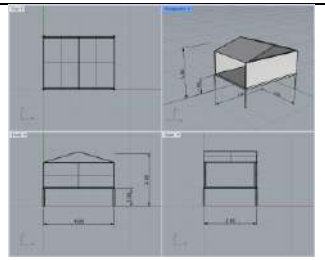
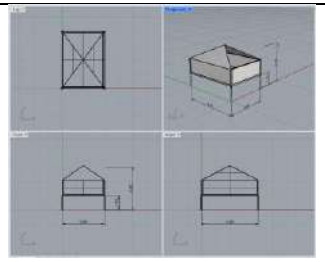
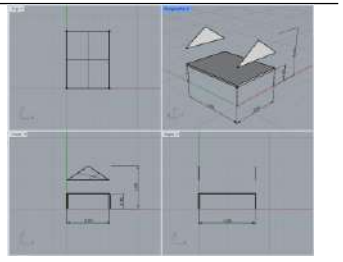
Yuan, Z., Lan, H., Zou, Q., & Zhao, J. (2024). 3D-PreMise: Can Large Language Models Generate 3D Shapes with Sharp Features and Parametric Control? *ArXiv Preprint ArXiv:2401.06437*. http://arxiv.org/abs/2401.06437

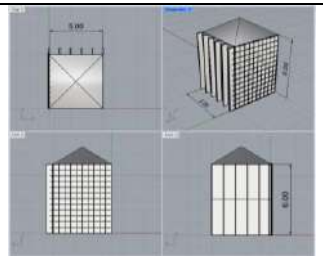
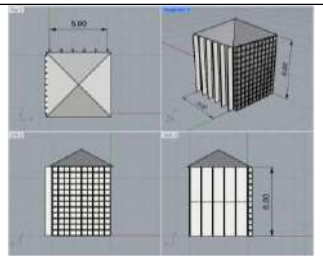
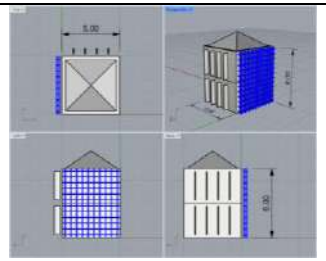
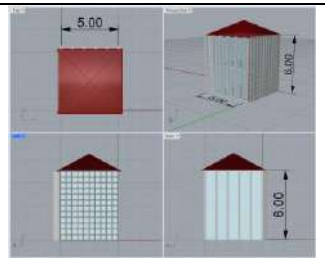
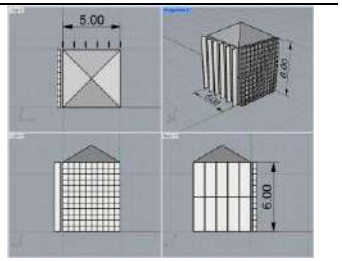
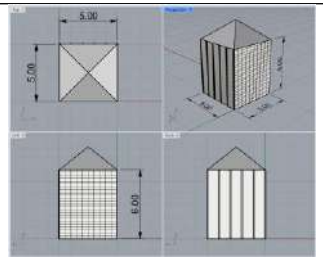
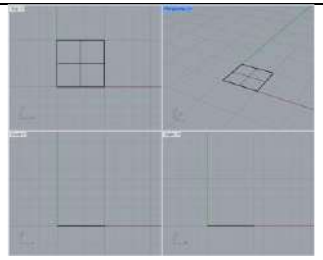
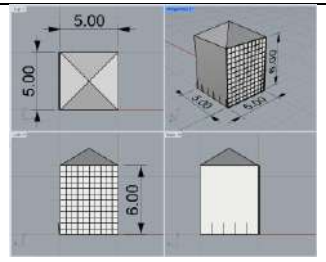
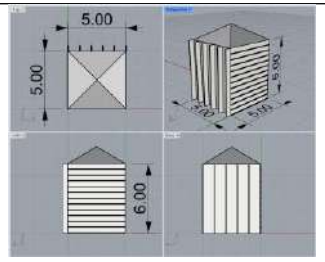
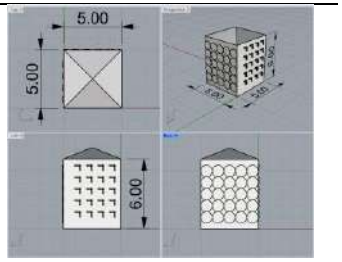
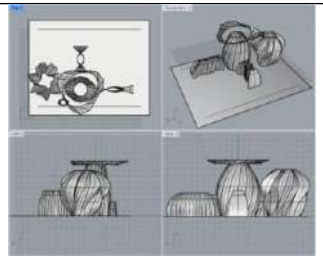
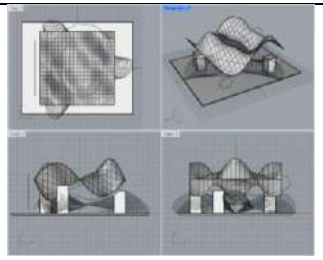
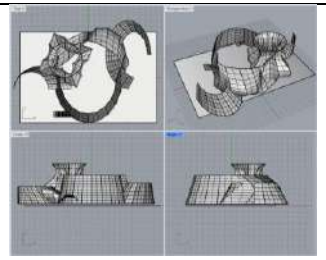
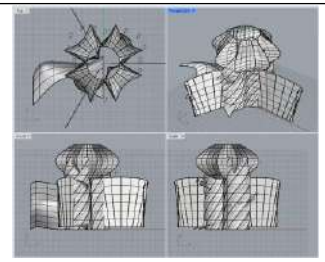
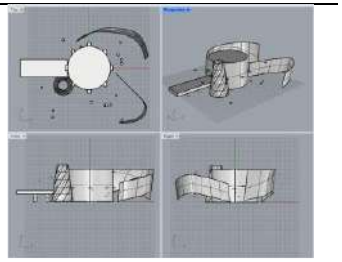
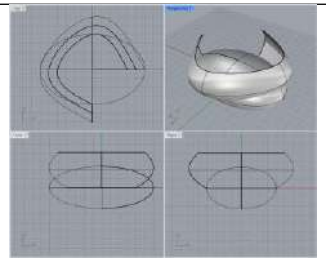
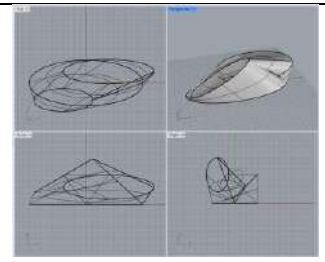
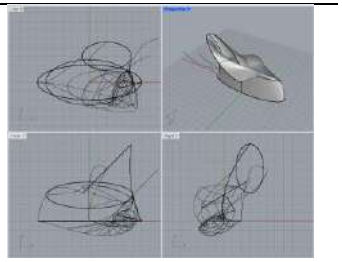
Zhang, J., Li, X., Wan, Z., Wang, C., & Liao, J. (2024). Text2NeRF: Text-Driven 3D Scene Generation with Neural Radiance Fields. *IEEE Transactions on Visualization and Computer Graphics*, *30*, 7749–7762. https://doi.org/10.1109/TVCG.2024.3361502

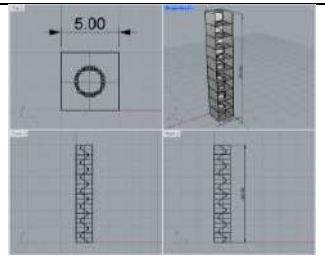
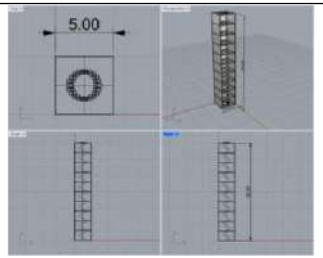
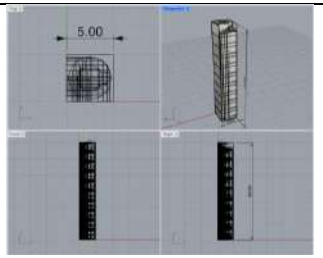
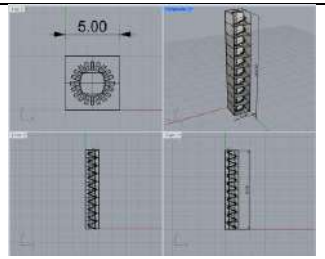
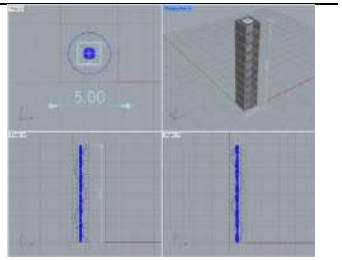
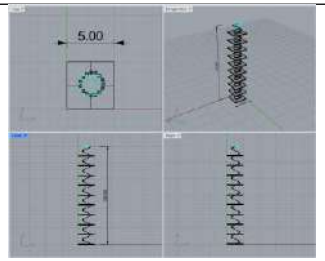
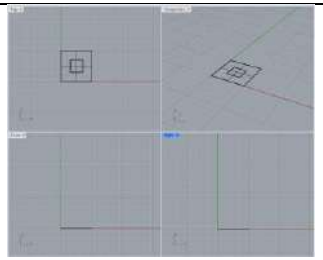
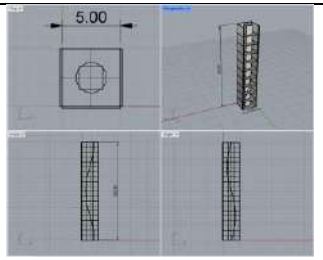
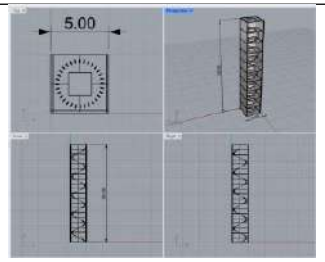
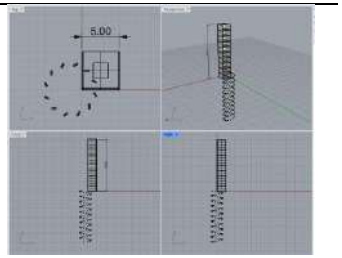
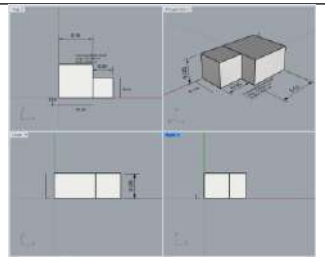
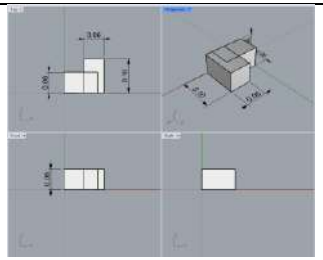
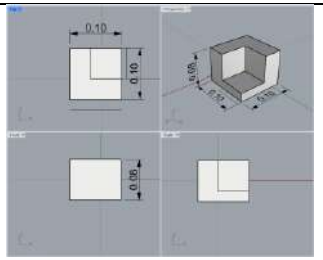
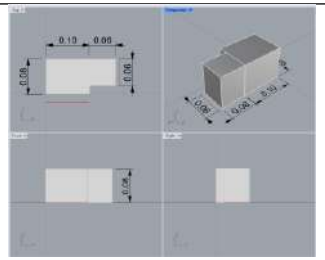
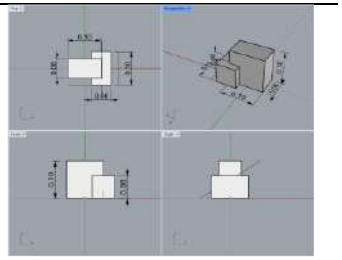
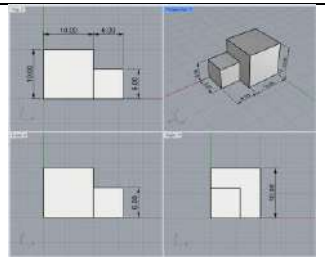
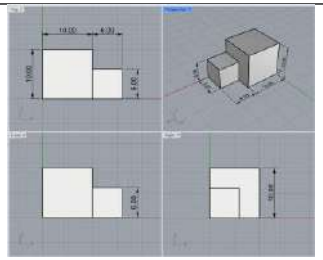
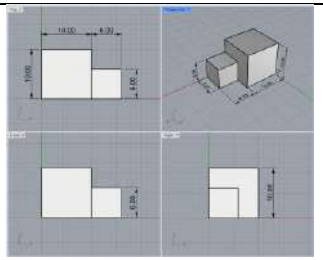
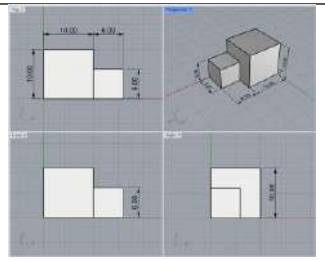
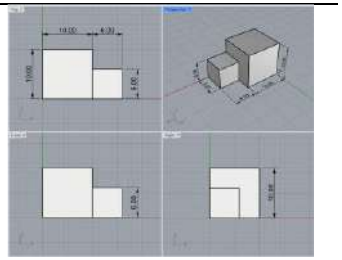
APPENDIX

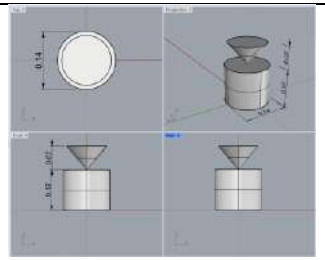
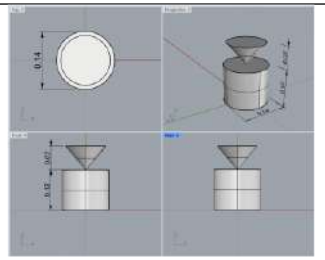
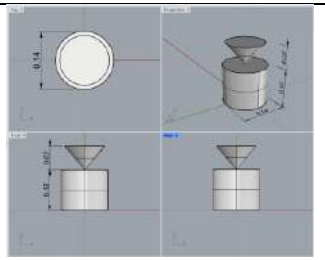
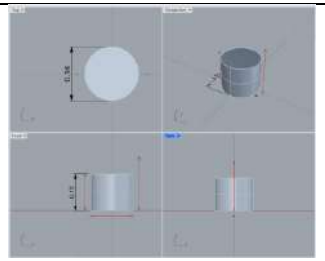
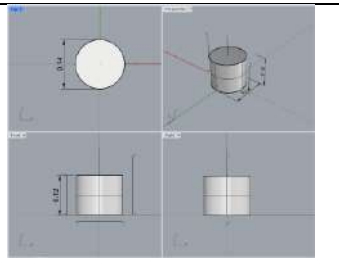
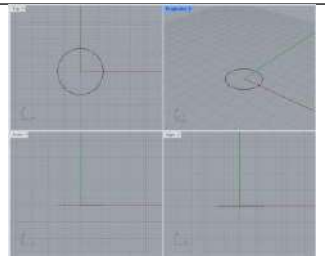
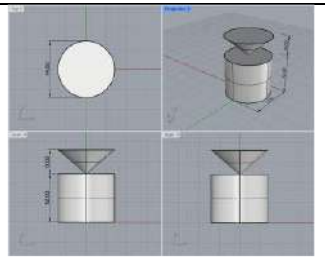
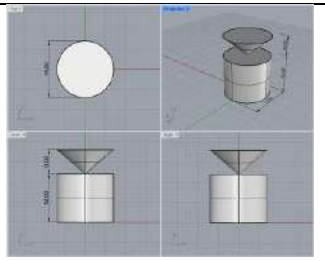
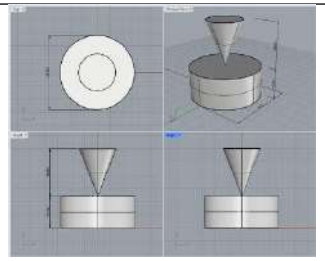
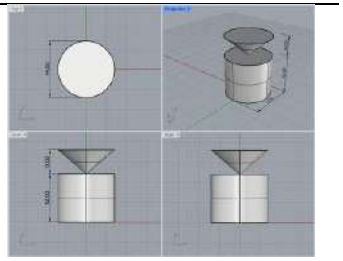
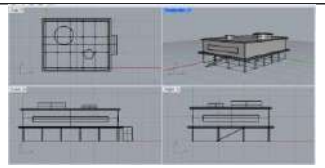
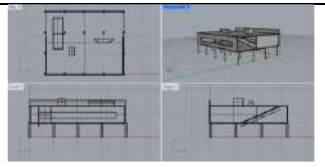
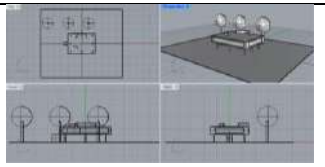
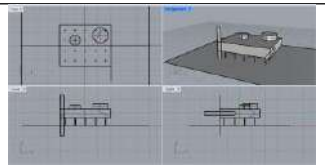
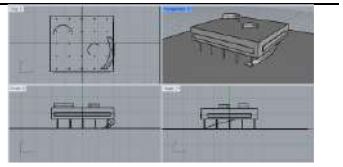
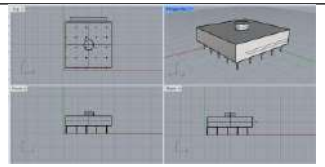
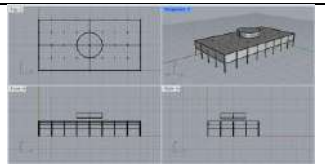
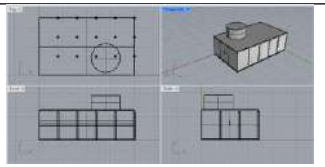
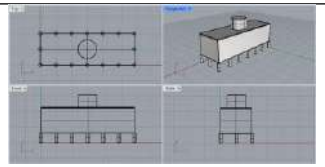
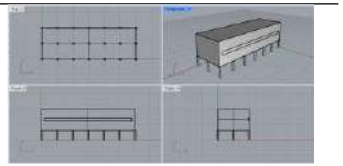
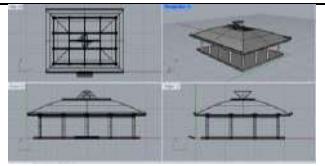
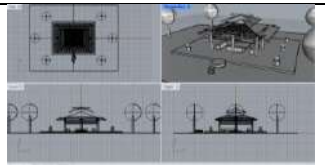
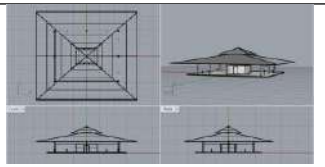
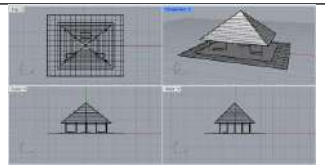
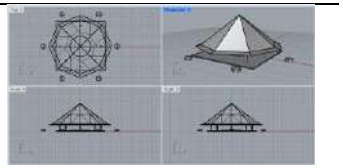
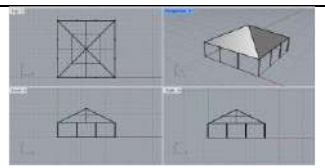
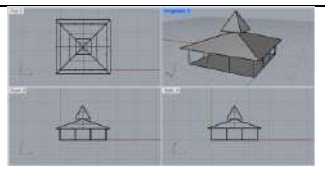
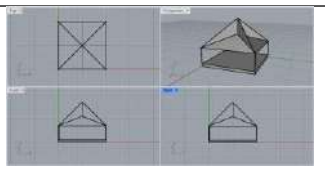
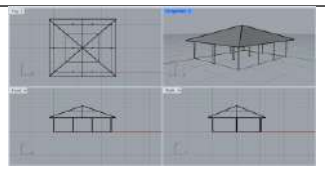
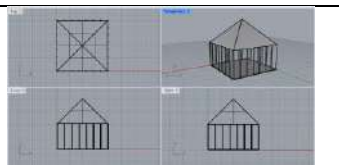
Prompt	Output 1	Output 2	Output 3	Output 4	Output 5
1 Claude					
1 ChatGPT					
2 Claude					
2 ChatGPT					

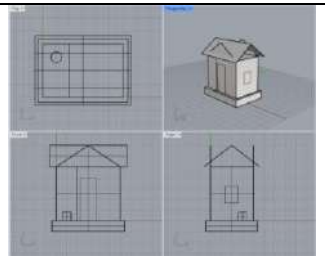
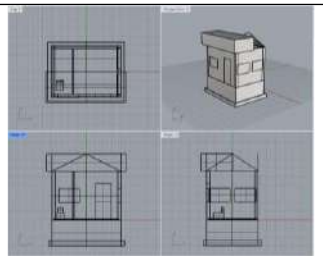
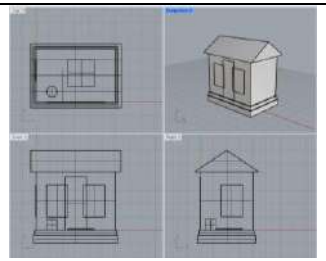
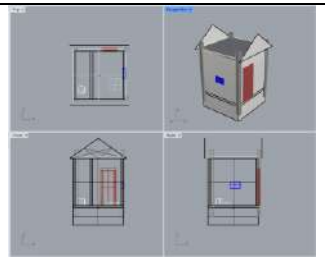
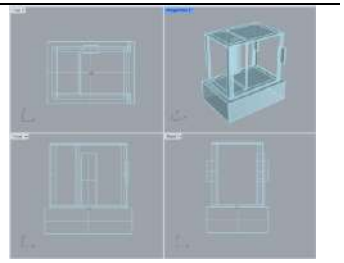
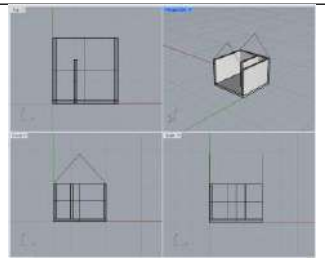
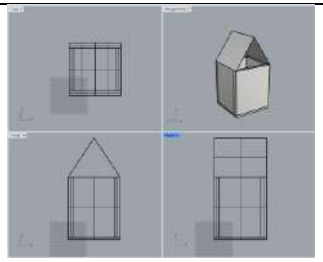
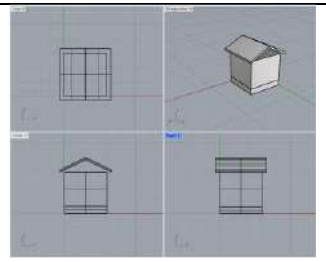
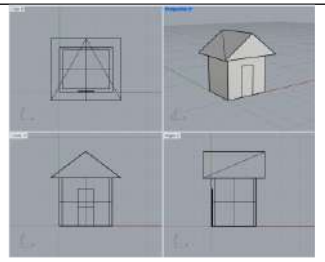
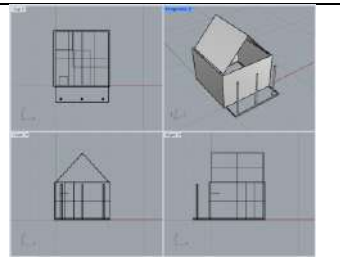
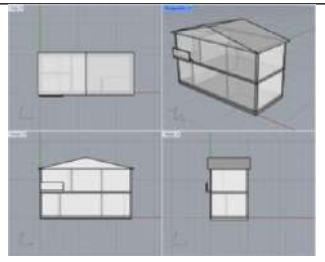
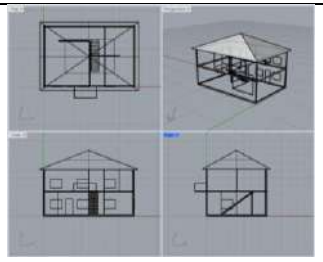
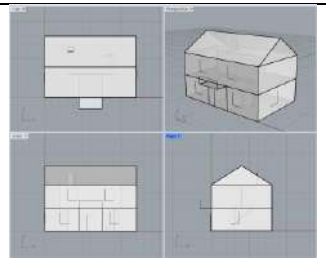
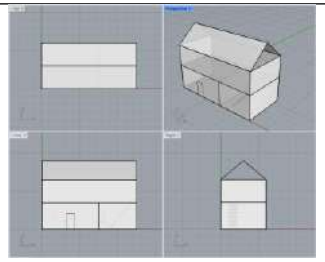
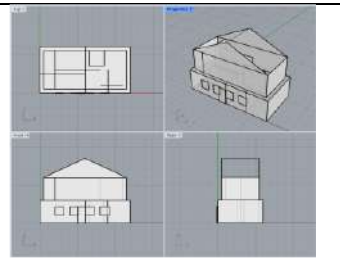
Prompt	Output 1	Output 2	Output 3	Output 4	Output 5
3 Claude					
3 ChatGPT					
4 Claude			NA		
4 ChatGPT		NA			

Prompt	Output 1	Output 2	Output 3	Output 4	Output 5
5 Claude					
5 ChatGPT					
6 Claude				NA	
6 ChatGPT					

Prompt	Output 1	Output 2	Output 3	Output 4	Output 5
7 Claude					
7 ChatGPT					
8 Claude					
8 ChatGPT	NA	NA			

Prompt	Output 1	Output 2	Output 3	Output 4	Output 5
9 Claude					
9 ChatGPT					
A Claude					
A ChatGPT					

Prompt	Output 1	Output 2	Output 3	Output 4	Output 5
B Claude					
B ChatGPT					
C Claude					
C ChatGPT					
D Claude					
D ChatGPT					

Prompt	Output 1	Output 2	Output 3	Output 4	Output 5
E Claude					
E ChatGPT					
F Claude					
F ChatGPT