iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 13 **Issue:** IV **Month of publication:** April 2025

DOI: https://doi.org/10.22214/ijraset.2025.68842

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering (Technology IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue IV Apr 2025- Available at www.ijraset.com

Text-To-Animation: Generating 3D Animation Using Textual Descriptions

Dr. A. Kousalya[1], Dhivya R[2], Kalaivani P[3], Revathi R[4], Dharani V[5]
[1]*Associate Professor,* [2, 3, 4, 5]*Student, Department of Artificial Intelligence and Data Science United Institute of Technology Coimbatore, India*

Abstract: ***The quick progress in generative AI Opened new doors in 3D animation by adding allowing automated animation generation from text description. This paper presents Text to Animation (TTA), a new system that fine-tunes the DeepSeek-Coder-5.7BMQA-Base model to convert natural language inputs into Blender Python Scripts. By incorporating Chainlit, the system provides real-time script generation and execution inside Blender with minimal human effort and improved user feedback. For performance rendering, animations are rendered on the cloud, maximizing computational effectiveness. The AI-based solution streamlines the animation pipeline for artists, teachers and creators by automating Blender scripting, hence reducing the technical barrier to producing high-quality 3D animations. It opens 3D content creation to a broader audience by minimizing the necessity of advanced technical skills. The model is fine-tuned with the DeepSeek-Coder-7.7BMQA-Base model with LoRA (Low-Rank Adaptation) for optimizing performance on tasks of generating Blender scripts. Possible users vary from training modules and educational simulation to gaming and cinematic previsualization. This article emphasizes how large language models transform creative industries by closing the gap between AI and 3D animation.Future research will concentrate on enhancing the model's comprehension of intricate motion dynamics and user interaction during the animation process.***

Keywords: Text-to-Animation, Blender, Model Fine-Tuning, DeepSeek-Coder-5.7BMQA-Ba se, AI-Generated Scripts, Chainlit, Cloud Rendering.

I. INTRODUCTION

The production of high-quality 3D animations normally demands considerable technical proficiency, especially in scripting environments like Blender's Python API. Although Blender provides rich procedural animation capabilities, creating these scripts manually tends to be complicated, time-consuming, and reserved for highly skilled developers or technical artists. Such a high barrier to entry reduces the possibility of extended creative exploration and innovation, particularly in users like educators, artists, and content creators. Emerging advancements in large language models (LLMs) have also brought forth new opportunities for automating these intricate tasks. Generally, these models can parse and generate executable code from natural language specifications, providing a more human-friendly method of user interaction with 3D software tools. Building on this ability, we propose an AI system that streamlines the process of creating animations immensely by translating text input into Blender Python scripts. Our suggested system, termed Text-to-Animation (TTA), fine-tunes the DeepSeek-Coder-5.7BMQA-Base model to translate user-input natural language instructions and produce related Blender scripts. Chainlit integration into the system enables real-time interaction such that users can inspect, edit, and run the created scripts within Blender's console without interruption. To improve performance and scalability, the system also includes cloud-based rendering, which guarantees that computationally demanding activities like animation rendering do not burden local hardware confines.

To enhance domain-specific precision and efficiency, we use Low-Rank Adaptation (LoRA) methods in the fine-tuning process. LoRA allow precise adaptation of the base model without much extra training overhead, improving it to generate accurate and optimized scripts for a vast range of animation contexts. Through the synergistic combination of strong language modeling, fine-tuning techniques, and cloud-based infrastructure, our system simplifies the procedural animation workflow and eliminates the historical impediments to adoption. The output workflow allows more users— educators, trainers, independent game developers, and filmmakers—to create high-quality, high-fidelity 3D content with reduced technical effort. This paper emphasizes the disruptive power of generative AI applied to procedural animation, demonstrating ways in which LLMs will be able to automate complex animations, enhance efficiency in production and make professional quality 3D content creation democratized. For that reason, it paves the way forward for future innovative human-AI collaboration, storytelling, and new immersive educational platforms.

II. REALTED WORK

Research on AI-based animation has focused on various techniques, such as text – to – image and video –to-video synthesis, procedural animation, and neural code generation. Generative models such as DALL·E, Stable Diffusion, and Video Diffusion have been shown to synthesize animations from text inputs. Procedural animation techniques use mathematical models to generate motions, while neural code generation uses largelanguage models (LLMs) togenerate Python scripts automatically for animation. Past work has explored the combination of AI and animation to support realism and computational efficiency.[1] exploresthe enhancementof texturefidelity and Coherence inanimation by integrating animation and computer vision.

In addition, [2] describes a survey on text – to – animation systems, outlining existing shortcomings. Unlike such methods, our approach is fine – tuning the "DeepSeek– Coder - 5.7 BMQA - Base" model for Blender script generation, integrating natural language processing, script execution, and rendering pipelines. This research aims to simplify text – to – animation pipelines to make it more accessible and efficient for creating 3D animations using Hugging Face LLMs, Blender, and Chainlit. Further, integration with cloud rendering allows for greater scalability and performance, removing the computational limitation of conventional methods.

III. METHODOLOGY

The suggested Text-to-Animation (TTA) system is developed to convert natural language descriptions to fully rendered 3D animations with the use of Blender. It combines high-level AI techniques with the use of the Blender Python API and cloud rendering to make 3D animation production more efficient and accessible. The system architecture involves four primary modules: Data Preparation, Model Training, Script Generation, and Rendering.

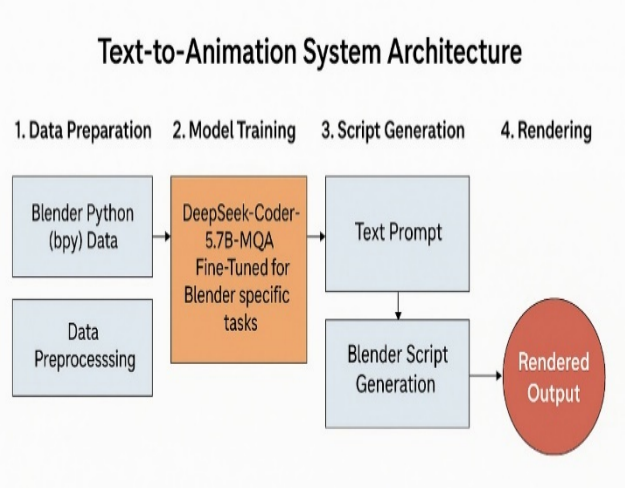


Figure 1: Text-to-Animation Architecture

A. Data Preparation

During the first phase, the system gathers a dataset that consists of Blender Python (bpy) scripts. These scripts have procedural logic, animation control, and object manipulation methods intrinsic to Blender's scripting interface. This data is subsequently preprocessed using cleaning, formatting, and tokenization to make it compatible with language model training. Preprocessing also involves removing duplicate or invalid code snippets, labeling script functionalities, and task segmentation for concentrated learning. This handpicked dataset constitutes the base knowledge the model employs to learn to create animation scripts.

B. Model Training

After preparing the data, the system then trains the language model. The system utilizes DeepSeek-Coder-5.7B-MQA as the underlying model. This is a strong code-generation model that is fine-tuned over the Blender-specific dataset to make it specialize in comprehending and creating Python scripts designed for Blender. Fine-tuning is further improved through LoRA (Low-Rank Adaptation), a parameter-efficient approach by which the model is allowed to learn the Blender-specific tasks with minimal computational overhead. This personalization allows the model to produce correct, working scripts from diverse and complicated text inputs.

C. Script Generation

At this stage, users interact with the system by entering natural language prompts, such as "Create a rotating cube with a bouncing motion." The fine-tuned model interprets the textual input and generates a corresponding Blender Python script. This script includes object creation, transformation, keyframe insertion, and animation logic required to realize the prompt. The script generation process is integrated into a Chainlit-based user interface, allowing real-time interaction, script display, and user feedback. This makes the tool highly accessible, even for users with no programming background.

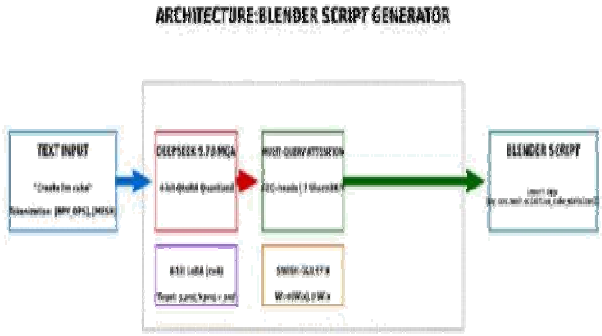


Figure 2: Model Architecture

D. Model Architecture

1) Text Input: The system starts by taking free-form natural language commands from the user, for example, "Make a 2-meter cube" or "Add a sphere of radius 1.5". These inputs are meant to simulate how a human would explain a 3D modeling operation verbally.

2) Tokenization: The input text is segmented into discrete tokens that map to Blender-specific operations and parameters. This step converts unstructured language into a machine-readable format.

3) Multi-Query Attention (MQA): The Multi-Query Attention (MQA) mechanism is a lightweight version of the conventional multi-head attention adopted in transformer architectures. With the regular multi-head attention scheme, each head comes with its own set of query, key, and values, which is computationally costly and memory-demanding, especially for big models. MQA responds by drastically decreasing the number of key and value matrices with the preservation of multiple query heads. Particularly, it has several independent query projections—to your example, 32 heads of queries—but a common set of key and value projections for all of these heads. This is achieved through this architecture choice which enables MQA to significantly minimize the computational overhead and memory consumption, making it highly efficient for large-scale language models during inference. Shared key-value mechanism ensures that the model retains critical contextual information from the input sequence without any redundant computation. Though the number of parameters and memory used are minimized, MQA retains the expressiveness of the model and achieves performance on par with full multi-head attention on most tasks. In the DeepSeek-Coder-5.7B-MQA model employed by your Text-to-Animation platform, MQA serves a crucial function by making inference more efficient without sacrificing Blender script quality. It allows the model to effectively process intricate user inputs—e.g., 3D modeling instructions—with haste and precision and is thus suitable for real-time or resource-scoped environments.

4) SWISH-GLU Feed-Forward Network (FPN): The SWISH-GLU Feed-Forward Network (FFN) is a sophisticated activation and gating mechanism that is used to increase the representational power of the model, specifically in learning intricate patterns—like those identified in Blender scripting operations. The architecture integrates two strong ideas: the Swish activation function and the Gated Linear Unit (GLU). Swish is given by the formula $x * \sigma(x)$, with $\sigma(x)$ representing the sigmoid function. In contrast to such conventional activation functions as ReLU, Swish facilitates smooth, non-monotonic activation so that gradients flow better during training. This enhances the model's capacity to pick up subtle variation in the input. GLU adds a gating mechanism so that portions of the input pass through selectively, enhancing feature selectivity and capacity to learn. When combined, the SWISH-GLU FFN can selectively activate and transform inputs in a non-linear and expressive manner.

Mathematically, it can be representedas:

SWISH-GLU(x) = (x * σ(x)) ⊗ (W_xx + b)

Here,⊗represents element-wise multiplication, and W_xx +b is a linear transformation on the input x. This two-part mechanism improves the performance of the model in decoding text prompts into high-level semantic patterns and translating them into correct Python commands for Blender operations.

In the architecture of DeepSeek-Coder-5.7B-MQA, the SWISH-GLU FFN is important to learn the compositional structure of Blender-related input texts so that the generated scripts are syntactically correct as well as contextually relevant.

5) Blender Script Output: The final generated Python script, ready for execution in Blender's runtime environment.

E. Rendering

Once the script is generated, it is executed within Blender to produce the 3D animation. To handle complex rendering tasks efficiently, the system leverages cloud rendering infrastructure. This enables high-performance rendering without burdening the user's local device, ensuring scalability and responsiveness. The final animation is outputted in a rendered format, ready for download or further use in educational, cinematic, or interactive applications.

IV. IMPLEMENTATION

The deployment of the Text-to-Blender Animation System includes fine-tuning an LLM for script generation, running the generated scripts in Blender, and rendering the animations in Google Colab.

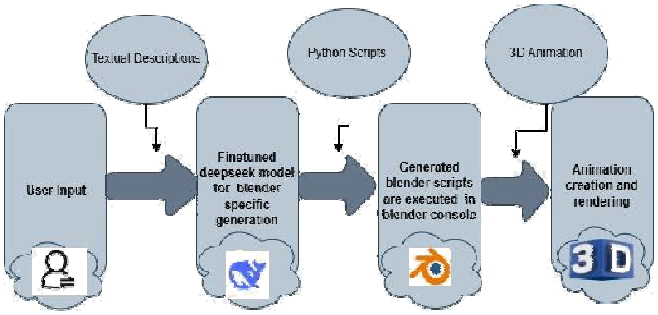


Figure 3: AI Driven Animation Creation Workflow

A. Training DeepSeek Model for Blender Specific Scripting

The DeepSeek–Coder-5.7bmqa - base model is fine – tuned on Blender API scripts using SpaCy - preprocessed documentation to extract relevant scripting patterns. Sliding Window tokenization (256-token windows,128-token stride) ensures efficient training. LoRa fine - tuning with 4 - bit quantization (QLoRA) optimizes GPU memory usage.

B. Real-Time Script Generation with Chainlit

The system employs Chainlit, a Python - based UI framework for LLMs, to generate the script in real -time. Natural language animation descriptions are fed by users, and they are processed by the fine-tuned model to produce Blender Python scripts. The generated scripts appear in the UI for preview purposes before being executed.

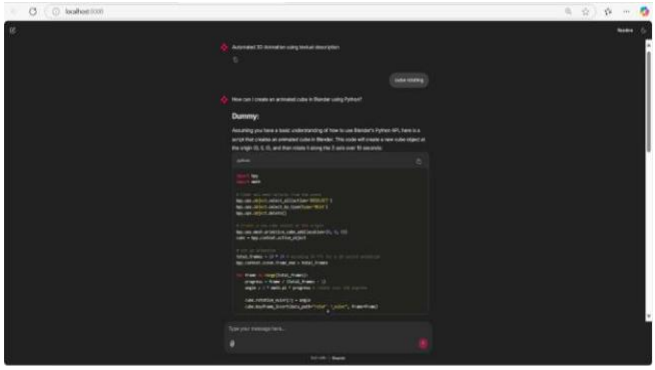


Figure 4: Real- Time Script Generation with Chainlit

C. Blender Script Execution and Animation Creation

The generated scripts are executed directly in Blender's Python console.
Object Creation – Generates 3d objects based on input prompts.
• Transformation & Animation – Applies transformations and keyframes.
Rendering – Uses Blender's internal rendering engine (Cycles or Eevee) for final animation output.

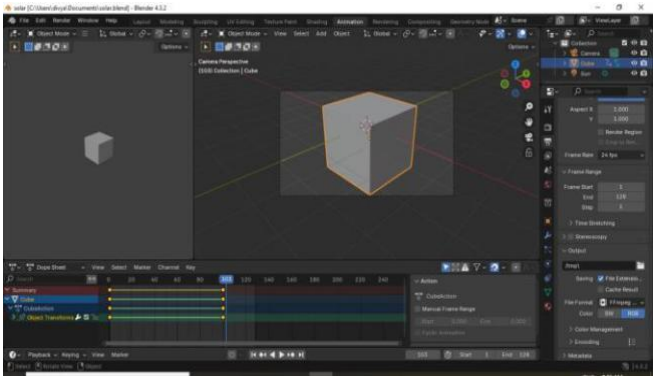


Figure 5: Blender Execution and Animation Creation

D. Cloud-Based Rendering in Google Colab

For cloud-based rendering, the system also integrates with Google Colab so that it can be run on GPU. The animation is rendered by Blender's MPEG-4 encoding, and the result is exported to Google Drive for downloading and playback.

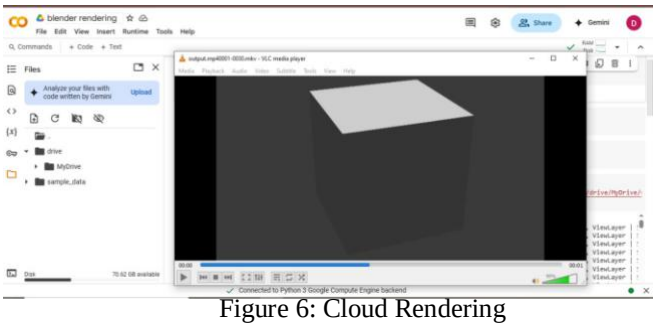


Figure 6: Cloud Rendering

V. RESULTS AND DISCUSSIONS

The Text-to-Blender Animation System proposed here offers a major leap toward connecting natural language processing and 3D animation. The system, with the help of a fine-tuned DeepSeek-Coder-5.7b-mqa-base model, is able to translate user-input text into corresponding Blender scripts. These scripts are automatically executed and rendered, yielding dynamic animations without any human coding involved. The automation makes the animation-making process easier, which can be accessed by anyone with no experience in scripting Blender.

A key component of this system is the integration with Chainlit, which provides an interactive, real-time preview of the scripts that are generated. This increases usability, as users can make adjustments immediately and see their animation output in real-time. The system's capacity to translate intricate textual commands into visual sequences is of tremendous worth in educational fields, particularly in medical education. Here, animations as a means to depict anatomical structures, physiological processes, or surgical procedures can greatly enhance understanding and retention.

Overall, this system represents the potential of generative AI in democratizing animation design. It eliminates technical hurdles, allows for creative exploration, and paves the way for new horizons in immersive learning experiences in many domains, revolutionizing how educational content is presented and absorbed.

VI. LIMITATIONS

Though the system shows promising outcomes, some limitations are there:

1) The accuracy of the scripts generated by Blender relies on input text quality and the training dataset.
2) The system now support simple3D transformations and animations but not complex physics – based simulations.
3) The speed of rendering is bound by hardware capacity, particularly when run in cloud - based platforms such as Google Colab.

VII. FUTURE WORK

Future improvements will focus on enhancing the system's animation complexity and scalability. Key areas for development include:

1) Expanding animation capabilities to support advanced motion dynamics, character rigging, and physics simulations in Blender.
2) Optimizing the model with domain-specific dataset to improve script accuracy for different industries, such as medical visualization, mechanical simulations, and educational content.
3) Integrating cloud-based GPU rendering for faster real-time animation previews within Chainlit UI.
4) Exploring multimodel AI approaches, such as incorporating image-based Guidance for animation generation, to improve precision and realism.

VIII. APPLICATIONS

The proposed Text-to-Blender Animation System has diverse applications across multiple domains, particularly in fields that require automated 3D visualization.

1) Medical Education: Enables the creation of surgical procedures, anatomical models, and medical simulations, enhancing the learning experience through interactive 3D animations.
2) STEM Education: Assists educators in generating physics - based simulations, mathematical visualizations, and engineering models for better conceptual understanding.
3) Virtual Training & Simulation:Useful for industrial training, flight simulations, and mechanical demonstrations, allowing users to visualize real-world processes dynamically.
4) Gaming & Animation Industry: Accelerates pre-visualization & prototyping by generating Blender Scriptsfor basic 3D object movements andtransformations.
5) Content Creation & StoryBoarding:Helpsfilmmakers andanimators generate quick scene layouts andmotion sequencesbased on text Descriptions.

IX. CONCLUSION

The envisioned Text-To-Animation System showcases the generative power of AI by turning text descriptions into fully rendered 3D animation. Through the fine -tuning of the DeepSeek-Coder-5.7bmqa-base model on Blender script data, the system streamlines the creation, execution, and rendering of Blender scripts. This streamlining greatly minimizes the complexity and time invested in animation creation, rendering it a useful technology for creative and technical applications. The incorporation of Chainlit as an interactive interface enables real - time preview of scripts, opening the system to users with little or no background in Blender scripting.

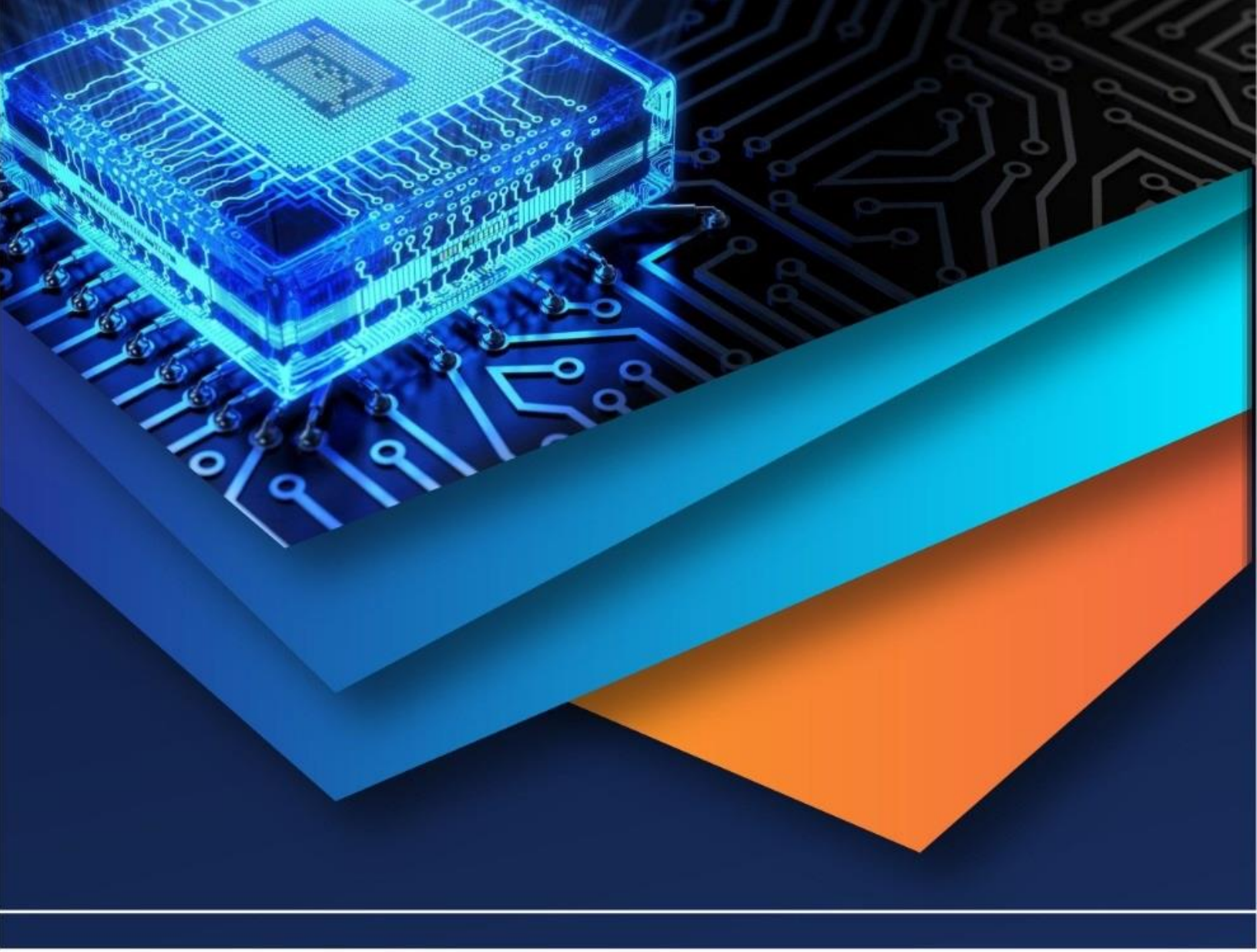
This friendly interface democratizes animation production, making it possible for anyone to turn abstract ideas into rich visual representations with ease. Such a feature is especially powerful in learning environments such as medical education, where interactive 3D animations can enhance conceptual understanding and learning outcomes.

Additionally, using Google Colab for GPU rendering makes it scalable and cost-effective without the requirement for Expensive local hardware. This is thus possible for educational institution, researchers & independent creators. Lastly, the Text-to-Animation System is a testament to the capability of generative AI to animation operations. Its capacity to automate animation design from text inputs has profound implications for new frontiers in education, research & creative sectors, where interactive visualizations are crucial to successful communication comprehension.

REFERENCES

[1] F. Zheng, Y Zhu, "Exploring the Fusion of Animation and Computer Vision for Enhanced Realism in Virtual Character Interaction," IEEE Access, vol. 12, pp. 194816– 194828, 2024, doi: 10.1109/ACCESS.2024.3519292.Available:https://doi.org/10.1109/ACCESS.2024.3519292

[2] N. Bouali and V. Cavalli-Sforza, "A Review of Text-to-Animation Systems," IEEE Access, vol. 11, pp. 86071– 86087, 2023, doi: 10.1109/ACCESS.2023.3304903. Available: https://doi.org/10.1109/ACCESS.2023.3304903
[3] Blender Foundation,BlenderPython API Documentation, Blender 4.3, 2024. Available: https://docs.blender.org/api/current.
[4] J. Jonathan Ho, T. Salimans, A. Gritsenko, W. Chan, M. Norouzi, and D. J. Fleet, "Video Diffusion Models," arXiv preprint, arXiv:2204.03458, 2022.Available:https://arxiv.org/abs/2204.03458
[5] Saharia, C., Chan,W., Saxena,S., Li, L., Whang, J.,Denton, E., SeyedGhasemipourS. K.,Ayan, B. K.,Mahdavi, S. S., Gontijo Lopes, R., Salimans, T., Ho, J.,Fleet, D. J., & Norouzi, M. (2022). Imagen: PhotorealisticText –to -ImageDiffusion Models with LargePretraine dModels. Available:https://imagen.research.google/
[6] U. Singer, A. Polyak, T. Hayes, X. Yin, J. An, S. Zhang, Q. Hu, H. Yang, O. Ashual, O. Gafni, D. Parikh, S. Gupta, and Y. Taigman, "Make-A-Video: Text-to-Video GenerationwithoutText-VideoData,"arXivpreprint,arXiv:2209.14792,2022.Available:https://arxiv.org/abs/2209.14792
[7] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-Resolution Image Synthesis with Latent Diffusion Models," arXiv preprint, arXiv:2112.10752, 2022. Available: https://arxiv.org/abs/2112.10752
[8] Z. Yang, J. Teng, W. Zheng, M. Ding, S. Huang, J. Xu, Y. Yang, W. Hong, X. Zhang, G. Feng, D. Yin, Y. Zhang, W. Wang, Y. Cheng, B. Xu, X. Gu, Y. Dong, and J. Tang, "CogVideoX: Text-to-Video Diffusion Models with AnExpert Transformer," arXiv preprint, arXiv:2408.06072,2025Available: https://arxiv.org/abs/2408.06072
[9] Z. Liu, Y. Meng, H. Ouyang, Y. Yu, B. Zhao, D. Cohen-Or, and H. Qu, "Dynamic Typography: Bringing Text to LifeviaVideoDiffusionPrior,"arXivpreprint,arXiv:2404.11614,2024.Available:https://arxiv.org/abs/2404.11614
[10] OpenAI.(2023).SORA:Self-Optimizing Reinforcement Agent. OpenAI. https://openai.com/
[11] M. Liu, Y. Liu, G. Krishnan, K. S. Bayer, and B. Zhou, "T2M-X: Learning Expressive Text-to-Motion Generationfrom Partially Annotated Data," arXiv preprint,arXiv:2409.13251,2024.Available:https://arxiv.org/abs/2409.13251
[12] P. Goel, K.-C. Wang, C. K. Liu, and K. Fatahalian, "Iterative Motion Editing with Natural Language," inProceedings of the ACM SIGGRAPH '24 Conference onComputer Graphics and Interactive Techniques, Jul. 2024,pp.1–9.Available:https://doi.org/10.1145/3641519.3657447



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY