Advanced International Journal for Research (AIJFR)
E-ISSN: 3048-7641 ● Website: www.aijfr.com ● Email: editor@aijfr.com

Cinematic Pre-Visualization System: Automated 3D Scene Generation from Natural Language Descriptions Using NLP and JSON-Based Scene Representation

Mrs. B. Dheepa[1], Monish Ram J[2], Praveen R[3], Shri Shivesh V S[4]

[1,2,3,4]Department of Information Technology, Sri Venkateswara College of Engineering
Affiliated to Anna University, Chennai, Tamil Nadu, India

ABSTRACT

The Cinematic Pre-Visualization System is an end-to-end pipeline that converts natural language scene descriptions into structured three-dimensional (3D) previsualization scenes in Blender and Unity. The system employs Natural Language Processing (NLP) techniques using the spaCy library to extract semantic scene elements including characters, objects, spatial relationships, lighting conditions, and camera directives from textual input. Extracted information is serialized into a platform-agnostic JSON intermediate representation, which drives automated 3D scene construction in Blender via the bpy Python API and in Unity via C# scripting. Experimental evaluation on twenty diverse cinematic scene descriptions demonstrates an average entity extraction accuracy of 87.5%, spatial relationship accuracy of 82.3%, lighting keyword detection of 94.1%, and camera directive extraction of 90.0%. The total pipeline latency from text input to rendered scene is under 6 seconds, validating the system's suitability for iterative use in real-world pre-production workflows. The system significantly reduces scene setup time and lowers the technical barrier for previsualization, offering a practical tool for the film, animation, and game production industries.

Keywords: cinematic previsualization; natural language processing; 3D scene generation; spaCy; Blender; Unity; JSON scene representation; pre-production automation

1. INTRODUCTION

Pre-visualization (previz) is the process of generating rough visual representations of film, animation, or game scenes prior to actual production. Traditional previz relies on manual 3D scene composition, storyboarding, and extensive collaboration between directors, writers, and technical artists — a process that is both time-consuming and resource-intensive. The Cinematic Pre-Visualization System addresses these challenges by providing an automated pipeline that converts plain-text scene descriptions into structured 3D scene data. In conventional film production, manual previsualization and scene blocking can take weeks or even months depending on scene complexity, whereas the proposed pipeline generates an initial previsualization in under six seconds, enabling rapid creative iteration.

The integration of NLP in modern creative tools is reshaping how the film and game industries approach early production. By parsing the semantic content of natural language sentences, extracting spatial relationships and environmental cues, and translating that information into 3D scene configurations, this project demonstrates that an end-to-end text-to-scene pipeline is feasible, practical, and highly effective within current computational constraints.

The primary motivation stems from the increasing demand for rapid iteration cycles during pre-production. Directors and producers must communicate scene compositions to technical teams quickly and accurately. Current tools require manual scene setup, delaying feedback loops and increasing production costs. Moreover, creative professionals are not always proficient with 3D software, creating a communication gap between written scripts and visual scene representation.

This paper makes the following primary contributions:

- An end-to-end NLP pipeline using spaCy for semantic extraction of cinematic scene elements from natural language descriptions.
- A platform-agnostic JSON intermediate scene representation schema enabling multi-platform 3D scene construction.
- Dual integration with Blender (bpy API) and Unity (C# scripting) driven by a unified JSON schema.
- Empirical evaluation on twenty diverse cinematic scene descriptions demonstrating competitive accuracy and sub-6-second pipeline latency.

2. RELATED WORK

Chang et al. [1] proposed a framework for generating 3D scene configurations from natural language descriptions using probabilistic grammars. Their system demonstrated that syntactic parsing of scene-related sentences could reliably extract object types and spatial constraints, though it was limited to a fixed vocabulary and did not generalize to creative cinematic language.

Ma et al. [6] introduced a neural network approach to text-to-scene synthesis using recurrent models to sequentially place objects in 2D layouts. While effective for simple scenes, the approach lacked support for 3D spatial reasoning and camera perspective modeling essential for cinematic previsualization.

The spaCy NLP library [2] has been extensively applied in information extraction tasks across legal document analysis, biomedical text mining, and news event extraction, demonstrating suitability for domain-specific structured extraction tasks such as cinematic script analysis.

Müller et al. [7] presented CityEngine, a procedural urban scene generation system driven by shape grammar rules. The concept of rule-driven scene construction from compact symbolic representations directly inspired the JSON-based scene specification approach in this work. Fisher et al. [3] proposed a probabilistic model for furniture layout encoding spatial relationship priors, conceptually related to the spatial_relations array in the proposed JSON schema.

Commercial previsualization tools such as Shotpro, FrameForge, and Cinedesigner provide manual scene construction capabilities but offer no natural language interface, leaving the gap between script text and

3D visual representation largely manual. Park et al. [8] demonstrated GAN-based photorealistic image synthesis from semantic label maps — a related but distinct approach operating on 2D image representations requiring large training datasets unsuitable for flexible 3D scene generation.

This work fills the identified gap by providing an end-to-end, text-driven previsualization pipeline with a platform-agnostic JSON intermediate representation, novel in its simultaneous targeting of both Blender and Unity as downstream 3D rendering platforms from a single unified description.

3. SYSTEM ARCHITECTURE AND DESIGN

A. Overall Architecture

The system is organized as a sequential four-stage processing pipeline: (1) Text Input Module, (2) NLP Processing Module, (3) JSON Generation Module, and (4) Scene Rendering Module. Data flows from raw natural language text to a rendered 3D scene. The architecture prioritizes modularity, extensibility, and platform independence — each module communicates with adjacent modules through clearly specified interfaces, enabling independent development and replacement of components.

B. NLP Processing Module

The NLP Processing Module is the core intelligence of the system. It employs spaCy's English language model (en_core_web_sm) to perform tokenization, part-of-speech tagging, dependency parsing, and named entity recognition. The module extracts six categories of scene information:

- Scene entities — nouns representing physical objects or characters.
- Spatial relationships — prepositional phrases indicating relative positions.
- Attribute descriptors — adjectives for color, size, and material.
- Action states — verbs describing entity behaviors.
- Lighting keywords — covering approximately 80 cinematic lighting descriptors.
- Camera directives — shot types and camera movements.

C. JSON Schema Design

Extracted scene information is encoded into a structured JSON document conforming to a custom cinematic scene schema. The schema defines a scene object containing a scene_id, description, timestamp, entities array, spatial_relations array, lighting object, and camera object. Each entity encodes a unique id, label, position coordinates (x, y, z), scale, rotation, material properties (color and texture), and an optional is_character flag. The lighting object specifies type, intensity, color, and direction. The camera object specifies type, position, target, and field-of-view value. The use of standard JSON ensures compatibility with any programming language capable of reading text files, making the representation truly platform-agnostic.

D. Blender Integration Module

The Blender Integration Module is a Python script executed within Blender's scripting environment using the bpy API. It reads the JSON file, clears the default scene, and iterates over the entities array to construct each scene element using appropriate primitive meshes — Cube for furniture/architecture, Sphere for globular objects, Cylinder for vertical structures, and Plane for ground surfaces. Materials are created dynamically using Blender's Principled BSDF shader with color values and roughness/metallic parameters

derived from the JSON entity attributes. The module is invoked headlessly via Blender's CLI, enabling batch processing and integration into automated production pipelines.

E. Unity Integration Module

The Unity Integration Module is implemented as a C# MonoBehaviour script that reads the JSON file at runtime using Unity's JsonUtility API. It instantiates prefab assets corresponding to identified scene entities, positions them according to JSON coordinates, configures lighting using Unity's Light component, and sets up the camera as specified. The result is a fully assembled Unity scene ready for real-time interactive preview, enabling directors and producers to navigate spatial compositions before committing to expensive physical set construction or detailed 3D modelling work.

4. IMPLEMENTATION

The entire NLP pipeline, JSON generation, and orchestration layer are implemented in Python 3.10. The Blender integration script is executed within Blender 3.6 LTS's embedded Python interpreter using the bpy module. The Unity integration is implemented in C# targeting Unity 2022.3 LTS. The command-line orchestration script main.py ties all components together.

The NLP pipeline is implemented in scene_parser.py, exposing a stateless, thread-safe parse_scene(text) function. The function processes input through the spaCy nlp pipeline, extracts nouns as candidate entities, uses dependency labels (nsubj, dobj, pobj) to determine entity roles, extracts adjective modifiers (amod dependency), identifies prepositional phrases for spatial relationships, and scans for lighting vocabulary and camera terminology. A heuristic spatial resolver maps prepositions (near, beside, behind, in front of, above) to relative coordinate offsets between entity pairs.

The json_generator.py module validates the structured dictionary against the defined schema before serialization, populates missing optional fields with sensible defaults, and produces human-readable JSON output. The output is written to a shared directory accessible to both Blender and Unity. The three modules are integrated through main.py, which orchestrates the pipeline and triggers Blender headless rendering via subprocess.

Table I: Hardware and Software Requirements

Component	Requirement
Processor	Intel Core i5 / AMD Equivalent (Recommended: i7+)
RAM	8 GB minimum, 16 GB recommended
GPU	Integrated Graphics (Recommended: NVIDIA GTX/RTX)
Python	3.10 or higher
spaCy	3.5 with en_core_web_sm model
Blender	3.6 LTS (bpy module)
Unity	2022.3 LTS
OS	Windows 11 / Ubuntu 20.04+

5. EXPERIMENTAL RESULTS AND DISCUSSION

The system was evaluated on a test set of twenty cinematic scene descriptions varying in complexity, vocabulary diversity, and scene type, covering interior scenes (taverns, laboratories, offices), exterior

scenes (beaches, markets, rooftops), and mixed environments. Descriptions ranged from 15 to 80 words. Ground-truth annotations were produced by two independent reviewers, and NLP output was compared against these annotations to compute extraction accuracy metrics.

Table II: Performance Evaluation Metrics

Metric	Value	Remarks
Entity Extraction Accuracy	**87.5%**	**Avg. across 20 cases**
Spatial Relation Extraction	**82.3%**	**Complex phrases reduced accuracy**
Lighting Keyword Detection	**94.1%**	**High-precision keyword list**
Camera Directive Extraction	**90.0%**	**Pattern matching approach**
NLP Processing Time	**0.32 sec**	**~50 word description**
Blender Scene Build Time	**4.7 sec avg**	**Up to 15 entities**
Unity Scene Load Time	**1.2 sec**	**Runtime loading**
Total Pipeline Latency	**< 6 sec**	**All tested scenes**
TC-06	**88.9%**	**Indoor laboratory scene with multiple objects**
TC-07	**91.4%**	**Outdoor marketplace with dense entities**
TC-08	**86.7%**	**Night street scene with lighting variations**
TC-09	**89.8%**	**Forest chase sequence with camera movement**

Table III: Comparison with Alternative Approaches

Approach	Accuracy	Context Understanding	Speed	Scalability
Keyword Matching	78%	Low	Fast	High
Pre-trained SBERT [11]	84%	Medium	Moderate	High
Proposed NLP+JSON System	**87.5%**	**High**	**Fast**	**High**

The system achieved an average entity extraction accuracy of 87.5%, spatial relationship extraction accuracy of 82.3%, lighting keyword detection of 94.1%, and camera directive extraction of 90.0%. Errors in entity extraction were primarily observed in complex nested prepositional phrases and metaphorical language. Spatial relationship errors were most frequent with three or more entities and chained spatial constraints.

The total pipeline latency from text input to rendered scene was under 6 seconds for all twenty tested scenes. NLP processing time averaged 0.32 seconds, Blender scene construction averaged 4.7 seconds, and Unity scene loading averaged 1.2 seconds. These results validate the system's practical suitability for iterative use during pre-production.

Current limitations include: (1) abstract or emotionally descriptive language is not translatable to 3D scene parameters; (2) primitive geometric shapes limit visual fidelity; (3) Unity prefab dictionary requires

manual curation for new entity types; and (4) the spatial relationship resolver applies heuristic offsets that may produce physically implausible arrangements for complex scenes.

A qualitative evaluation was also conducted during the mini-project review. An external industry reviewer observed that the proposed system has strong commercial potential and suggested that it could be presented to film production houses and studios as a practical pre-production tool, highlighting its ability to significantly reduce the time and effort required for early scene visualization.

6. CONCLUSION AND FUTURE WORK

This paper presented the Cinematic Pre-Visualization System, an end-to-end pipeline converting natural language scene descriptions into structured 3D previsualization scenes in Blender and Unity. The system demonstrates that NLP techniques — specifically spaCy dependency parsing and named entity recognition — can effectively extract semantic content of cinematic text descriptions with 87.5% average entity extraction accuracy. The JSON-based intermediate representation proved a robust, flexible interface enabling platform-agnostic scene specification, and the dual Blender/Unity integration produced coherent 3D scene layouts within 6 seconds of text input.

Future work will focus on: (1) integrating a large language model (LLM) for richer semantic understanding of abstract and metaphorical descriptions; (2) incorporating 3D asset library search (Sketchfab, Polyhaven APIs) and generative 3D object creation to automatically generate complete objects instead of relying only on primitive shapes such as cubes and spheres; (3) developing a browser-based graphical user interface with an embedded WebGL viewer; (4) extending support to sequential scene descriptions for full-sequence animatic generation from complete screenplays; (5) enabling real-time collaborative previsualization through a web-based shared viewer; and (6) replacing heuristic coordinate assignment with a machine learning-based spatial layout optimizer trained on professionally composed 3D scenes. This enhancement will enable the generation of complete scene objects directly from textual descriptions.

REFERENCES

1. A. Chang, W. Monroe, M. Savva, C. Potts and C. D. Manning, "Text to 3D Scene Generation with Rich Lexical Grounding," Proc. COLING 2016, pp. 2665–2677.
2. Explosion AI, "spaCy: Industrial-Strength Natural Language Processing," 2023. [Online]. Available: https://spacy.io
3. M. Fisher, D. Ritchie, M. Savva, T. Funkhouser and P. Hanrahan, "Example-based Synthesis of 3D Object Arrangements," ACM Trans. Graphics, vol. 31, no. 6, pp. 135:1–135:11, 2012.
4. M. Hendrikx, S. Meijer, J. Van Der Velden and A. Iosup, "Procedural Content Generation for Games: A Survey," ACM Trans. Multimedia Comput., vol. 9, no. 1, pp. 1:1–1:22, 2013.
5. S. Louchart and R. Aylett, "Narrative Theory and Emergent Interactive Narrative," Int. J. Cont. Eng. Educ., vol. 14, no. 6, pp. 506–518, 2004.
6. C. Ma, Z. Li and A. X. Chang, "Language-Driven Synthesis of 3D Scenes from Scene Databases," ACM Trans. Graphics, vol. 37, no. 6, pp. 212:1–212:16, 2018.
7. P. Müller, P. Wonka, S. Haegler, A. Ulmer and L. Van Gool, "Procedural Modeling of Buildings," ACM SIGGRAPH 2006, pp. 614–623.

8. T. Park, M. Y. Liu, T. C. Wang and J. Y. Zhu, "Semantic Image Synthesis with Spatially-Adaptive Normalization," Proc. IEEE/CVF CVPR, pp. 2337–2346, 2019.
9. Blender Foundation, "Blender Python API Documentation," 2023. [Online]. Available: https://docs.blender.org/api/current
10. Unity Technologies, "Unity Scripting API," 2023. [Online]. Available: https://docs.unity3d.com/ScriptReference
11. N. Reimers and I. Gurevych, "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks," Proc. EMNLP 2019, pp. 3982–3992.
12. S. Antol et al., "VQA: Visual Question Answering," Proc. IEEE ICCV, pp. 2425–2433, 2015.
13. A. Radford et al., "Learning Transferable Visual Models From Natural Language Supervision," Proc. ICML, 2021.
14. A. Vaswani et al., "Attention Is All You Need," Proc. NeurIPS, 2017.
15. Khronos Group, "glTF 2.0 Specification," 2023.