Patent Application Materials

System and Method for Generating Interactive 3D Scenes from Natural Language Prompts Using a Constrained Asset Framework

---

1. PROVISIONAL PATENT APPLICATION COVER SHEET

Title of Invention: System and Method for Generating Interactive 3D Scenes from Natural Language Prompts Using a Constrained Asset Framework

Inventor(s): Javier Anta Callersten

Citizenship: Spain

Residence: Avenida de la Osa Mayor 15 bis, 28023, Madrid, Spain

Correspondence Address: Avenida de la Osa Mayor 15 bis, 28023, Madrid, Spain

Filing Date: 22nd July 2025

Application Number: [To be assigned by USPTO]

---

2. SPECIFICATION

This provisional patent application is filed under 35 U.S.C. §111(b).

Applicant requests a foreign filing license under 37 CFR 5.2.

CROSS-REFERENCE TO RELATED APPLICATIONS

Not Applicable.

STATEMENT REGARDING FEDERALLY SPONSORED RESEARCH

Not Applicable.

BACKGROUND OF THE INVENTION

Field of the Invention The present invention relates to computer-implemented systems and methods for generating three-dimensional (3D) scenes, and more particularly to systems that convert natural language inputs into interactive 3D environments using artificial intelligence constrained by predefined asset libraries.

Description of Related Art Current approaches to 3D scene generation suffer from several limitations:

1. Traditional 3D modeling requires specialized technical skills and significant time investment
2. Existing AI-based 3D generation systems produce inconsistent quality and unpredictable outputs
3. Text-to-3D systems often generate geometries that are not optimized for real-time rendering
4. Current procedural generation lacks semantic understanding of user intent

Problems with Prior Art

- Unconstrained AI generation leads to hallucinated or non-renderable assets
- No effective bridge between natural language understanding and practical 3D scene assembly
- Lack of quality control in automated 3D generation systems
- Poor performance in web-based and mobile rendering environments

SUMMARY OF THE INVENTION

The present invention provides a system and method that enables non-technical users to generate interactive 3D scenes from natural language prompts while maintaining consistent quality and renderability through a constrained asset framework.

Key Advantages:

1. Guarantees renderable output by constraining to pre-validated assets
2. Enables rapid scene generation (seconds vs. hours)
3. Maintains consistent visual quality across all generated scenes
4. Supports real-time exploration in web browsers and VR headsets
5. Prevents AI hallucination through structured validation

DETAILED DESCRIPTION OF THE INVENTION

System Architecture Overview

The invention comprises four primary subsystems:

1. Natural Language Processing Module (100)

- Receives text or image input from user
- Extracts semantic scene elements (objects, spatial relationships, lighting, mood)
- Converts natural language into structured scene specifications
- Interfaces with Large Language Models (LLMs) via API

2. Constrained Asset Selection Engine (200)

- Maintains database of pre-validated 3D assets with metadata tags
- Maps semantic requirements to available assets
- Enforces composition rules and budgets
- Implements variant selection for diversity

3. Scene Layout Generator (300)

- Determines optimal spatial arrangement of selected assets
- Applies physics constraints and collision detection
- Generates transformation matrices for each object
- Outputs structured scene definition (JSON/XML)

4. Runtime Rendering System (400)

- Loads and instantiates 3D assets in web browser
- Applies transformations and lighting
- Enables real-time navigation and interaction
- Supports WebXR for VR/AR viewing

Detailed Process Flow:

Step 1: User inputs natural language prompt Example: "Create a cozy coffee shop with vintage furniture and warm lighting"

Step 2: NLP module extracts key elements:

- Environment type: interior, commercial
- Objects needed: furniture, coffee equipment
- Atmosphere: cozy, vintage, warm

Step 3: Constraint engine queries asset database:

- Filters assets by tags: "furniture", "vintage", "coffee_shop"
- Checks availability and compatibility
- Applies budgeting rules (max 20 objects, 5MB total)

Step 4: Layout generator creates scene structure, for example:


```
{
 "scene": {
  "environment": "interior_coffeeshop_01",
  "lighting": "warm_ambient",
  "objects": [
   {
    "id": "vintage_chair_02",
    "position": [2.5, 0, -1.2],
    "rotation": [0, 45, 0],
    "scale": [1, 1, 1]
   },
   {
    "id": "wooden_table_round_01",
    "position": [2.3, 0, -1.0],
    "rotation": [0, 0, 0],
    "scale": [1.2, 1, 1.2]
   },
   {
    "id": "retro_floor_lamp_01",
    "position": [1.8, 0, -1.5],
    "rotation": [0, 90, 0],
    "scale": [1, 1, 1],
    "lighting_override": {
     "type": "point",
     "intensity": 0.8,
     "color": "#FFD8A6"
    }
   }
  ]
 }
}
```


Step 5: Runtime system renders interactive scene

NOVEL TECHNICAL FEATURES:

1. **Semantic-to-Asset Mapping Algorithm**
   - Multi-tier matching system with fuzzy logic
   - Contextual understanding of object relationships
   - Fallback strategies for missing assets
2. **Procedural Budgeting System**
   - Dynamic allocation based on scene complexity
   - Performance-aware asset selection
   - Mobile/desktop optimization paths

3. **Validation Pipeline**
    - Pre-flight checks for scene viability
    - Asset dependency resolution
    - Error recovery mechanisms
4. **Composable Asset Framework**
    - Hierarchical asset grouping
    - Spawn patterns and templates
    - Variant instancing framework
5. **Real-time Performance Optimization**
    - Progressive asset loading based on viewport visibility
    - Automatic hiding of objects outside camera view (frustum culling)
    - Dynamic level-of-detail (LOD) switching based on camera distance

COMMERCIAL ADVANTAGES

One embodiment of the invention provides significant commercial benefits including:

- Significant reduction in 3D content creation time
- No specialized training required for users
- Guaranteed mobile and VR compatibility
- Scalable cloud-based architecture

CLAIMS

Claim 1: A computer-implemented method for generating interactive 3D scenes, comprising:

- receiving a natural language prompt describing a desired 3D scene;
- extracting semantic scene elements including spatial relationships, object types, and stylistic attributes using natural language processing;
- constraining asset selection exclusively to a pre-validated library of 3D assets, wherein each asset includes performance metadata comprising polygon count, texture memory requirements, and draw call impact metrics;
- applying a constraint satisfaction framework that enforces performance budgets, geometric compatibility rules, and semantic coherence validation;
- generating a spatial layout using physics-based placement with collision and stability constraints;
- outputting a structured scene definition optimized for real-time web-based rendering in browsers without plugin requirements.

Claim 2: The method of claim 1, further comprising:

- enforcing constraints on asset selection based on performance budgets;
- validating compatibility between selected assets;
- applying procedural variations to enhance visual diversity.

Claim 3: The method of claim 1, wherein the asset selection comprises a fuzzy matching process that:

- evaluates the similarity between extracted semantic elements and asset metadata tags using a weighted scoring model;
- assigns relevance scores based on match types, including exact matches, synonyms, and contextually related terms, with respective weights;
- incorporates embedding-based similarity measures using pre-trained vector models for multi-language and domain-specific disambiguation; and
- ranks candidate assets by composite relevance scores to identify the most appropriate matches for scene generation.

Claim 4: A system comprising:

- a natural language processor (100);
- a constraint engine (200) with budget enforcement and substitution logic;
- a layout generator (300) with AI-based placement and physics rules;
- a renderer (400) supporting real-time WebGL/WebXR output.

Claim 5: The system of claim 4, further comprising a validation module for geometry overlap detection, semantic compatibility checking, and performance constraint enforcement.

Claim 6: The system of claim 4, further comprising an asset variation framework using interchangeable templates, materials, and shader configurations for procedural diversity.

Claim 7: The system of claim 4, further comprising a user-editable scene composition layer allowing constrained repositioning of objects and live feedback on layout validity.

Claim 8: The constraint engine of claim 4 includes:

- hierarchical categorization and tags;
- performance-aware filtering;
- intelligent substitution based on a visual similarity graph.

Claim 9: A medium storing code to:

- convert prompts into structured definitions;
- select assets within budget and compatibility thresholds;
- output optimized 3D scenes from predefined resources.

Claim 10: The method of claim 1 further supports multi-turn prompt refinement using conversational memory, including tracking previous user instructions and applying weighted prioritization to newly added semantic elements.

Claim 11: The system of claim 4 includes a telemetry engine to track:

- asset load times, draw call density, FPS trends, and bottlenecks;
- automatic optimizations based on historical data.

Claim 12: The method of claim 1 supports procedural variation via:

- texture swap with UV integrity;
- real-time HSL color tuning;
- shader-modified wear effects, including dirt, scratch, or fade patterns, generated via GLSL-based procedural noise functions mapped to asset UV coordinates.

Claim 13: The system of claim 4 includes a multi-tier caching mechanism that uses memory and disk layers, and employs usage-based or time-based invalidation strategies to maintain freshness and performance efficiency comprising:

- a multi-tier cache hierarchy with memory and disk-based storage;
- predictive pre-loading based on semantic similarity to current scene requirements;
- tracking of asset usage frequency and automatically promoting frequently used assets to faster cache tiers;
- synchronized cache state across multiple user sessions using versioned manifests.

Claim 14: The system of claim 4, wherein the rendering system includes comprehensive WebXR support comprising:

- automatic scene scale adjustment for room-scale virtual reality experiences;
- stereoscopic rendering optimization with foveated rendering support;
- hand tracking integration for controller-free interaction with generated scenes;
- augmented reality mode with real-world occlusion and lighting estimation.

Claim 15: The method of claim 1, further comprising a scene sharing system that:

- generates unique URLs encoding compressed scene state without exposing asset paths;
- implements view-only and interactive permission modes for shared scenes;
- tracks engagement metrics including view count, interaction time, and user modifications;
- enables collaborative editing with real-time synchronization between multiple users.

Claim 16: The system of any of claims 5 through 7, wherein the validation module performs multi-stage compatibility checking comprising:

- geometric intersection testing to prevent object overlap and z-fighting;
- semantic compatibility verification to ensure that object combinations adhere to plausible functional and spatial roles (e.g. tables under cups, chairs not intersecting walls);
- performance budget validation with graceful degradation strategies; and
- platform-specific compliance checking for desktop, mobile, and VR rendering targets.

Claim 17: The method of claim 3, wherein the fuzzy matching system employs:

- multi-language synonym recognition using embedded word vectors;
- contextual disambiguation based on co-occurrence patterns in the scene;
- industry-specific terminology mapping for specialized domains;
- confidence scoring with transparent reporting of match quality to guide asset selection.

Claim 18: The system of claim 4 further comprises a performance monitoring module that uses browser-based profiling tools (e.g., WebGL timer queries or WebXR diagnostic APIs) to measure frame rendering stages with sub-millisecond precision and detect performance bottlenecks:

- tracks frame timing with sub-millisecond precision across rendering pipeline stages;
- identifies performance bottlenecks through automated profiling;
- suggests optimization strategies including LOD adjustments and asset substitutions;
- maintains historical performance data for trend analysis and regression detection.

Claim 19: The method of claim 1, wherein the AI-driven spatial placement incorporates:

- physics-based constraints including gravity, collision detection, and stability analysis;
- learned placement patterns from analyzing professionally designed 3D scenes;
- context-sensitive object arrangement rules, informed by curated placement datasets or user locale settings;
- iterative refinement through constraint satisfaction solving with backtracking.

Claim 20: The system of claim 6, wherein the fallback mechanisms implement intelligent asset substitution by:

- maintaining a similarity graph between assets based on visual and functional properties;
- calculating substitution impact scores considering both visual fidelity and performance;
- providing user-configurable preference weights between quality and performance;
- generating explanations for substitution decisions to maintain user trust and control.

Claim 21: The method of any of claims 1 through 3, further comprising post-generation editing capabilities, wherein the system provides:

- object-level manipulation with constrained transformations to preserve physical and spatial coherence;
- real-time lighting adjustment with physically-based rendering previews;
- material editing functions that propagate changes across all instances of a given asset; and
- versioning support including scene history tracking, reversible modifications, and branching support for exploratory scene variants.

Claim 22: A distributed system implementation of claim 4, wherein:

- the natural language processing module operates on edge devices for privacy and low latency;
- the constraint engine executes on scalable cloud infrastructure with dynamic resource allocation;
- the asset library utilizes content delivery networks with geographic distribution;
- the rendering system supports both server-side rendering with pixel streaming and client-side WebGL rendering based on available bandwidth and device capabilities.

ABSTRACT

A system and method for generating interactive 3D scenes from natural language prompts using artificial intelligence constrained by a predefined asset library. The system parses user input to extract semantic elements, maps those elements to pre-validated 3D assets using a fuzzy matching engine with scoring and disambiguation logic, and generates an optimized spatial layout that adheres to performance and platform constraints. The resulting scene is rendered in real time using a web-based engine supporting WebGL and WebXR. By constraining AI output to known, renderable assets and enforcing composition rules, the system ensures consistent quality and predictable performance. This enables immediate usability across desktop, mobile, and VR platforms. This enables non-technical users to create high-quality 3D environments in seconds.

---

3. DRAWINGS

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 shows a system architecture overview of the constrained 3D scene generation system, showing NLP Module (100), Constraint Engine (200), Layout Generator (300), and Renderer (400)

Figure 2 shows a process flow diagram from user input to rendered output, and illustrates step-by-step transformation from text input to rendered scene

Figure 3 shows the hierarchical structure of the constrained asset library, with tagging and variant logic

Figure 4 shows a sample transformation from natural language prompt input to generated scene JSON

Figure 5 shows the constraint engine decision tree logic, with budget validation and fallback logic.

Figure 6 shows the rendering pipeline from scene definition to multi-device WebGL/WebXR rendering.

---

4. INVENTOR'S DECLARATION

I hereby declare that:

- I am the original and sole/joint inventor of the subject matter claimed
- I have reviewed and understand the contents of this application
- I acknowledge the duty to disclose all information material to patentability

Signature: /Javier Anta Callersten/

Date: July 22nd 2025

---

5. INFORMATION DISCLOSURE STATEMENT (IDS)

Potentially Relevant Prior Art Analysis:

1. US10665030B1 - "Visualizing Natural Language Through 3D Scenes in Augmented Reality" (Google LLC, 2020)
    - Distinguishable: Google's patent focuses on AR deployment using probabilistic scene augmentation methods. Our invention differs by: (a) implementing a structured constraint satisfaction framework rather than probabilistic methods, (b) targeting web-based deployment optimized for browser rendering, (c) providing interactive post-generation editing capabilities, and (d) enforcing performance budgets with guaranteed renderability.
2. GraphDreamer: Compositional 3D Scene Synthesis from Scene Graphs (Max Planck, 2024)
    - Distinguishable: Academic research using scene graphs for geometry generation. Our system differs by: (a) direct natural language processing without intermediate scene graph conversion, (b) constraint-based asset selection from pre-validated libraries, (c) real-time interactive capabilities, and (d) production-ready web deployment architecture.
3. Unity ProBuilder & Unreal PCG Framework
    - Distinguishable: Require technical expertise with node-based systems, lack natural language interfaces, and focus on editor-based rather than web-deployed generation.
4. NVIDIA Omniverse USD Search API

    - Distinguishable: Implements natural language asset search but lacks complete scene generation pipeline, constraint satisfaction framework, and interactive manipulation capabilities
5. Luma AI Genie & TRELLIS (Microsoft)
    - Distinguishable: Focus on individual 3D object generation rather than complete scene composition, lack constraint frameworks for quality assurance, and don't provide interactive post-generation editing.
6. NVIDIA's GET3D
    - Distinguishable: GET3D is focused on mesh generation from 2D images or latent space sampling and does not produce scene layouts or interactive environments. Unlike our system, it does not use natural language input or enforce constraints on asset quality, spatial validity, or platform-specific rendering performance.
7. Published Paper: "SceneGPT: Language-Driven 3D Scene Generation" (2023)
    - Distinguishable: SceneGPT performs language-driven 3D scene generation by directly generating novel geometry using generative models, but lacks a constrained asset framework. It does not ensure renderability, asset compatibility, or real-time performance, leading to unpredictable outputs that are unsuitable for production environments.

Figure 1: System Architecture Overview

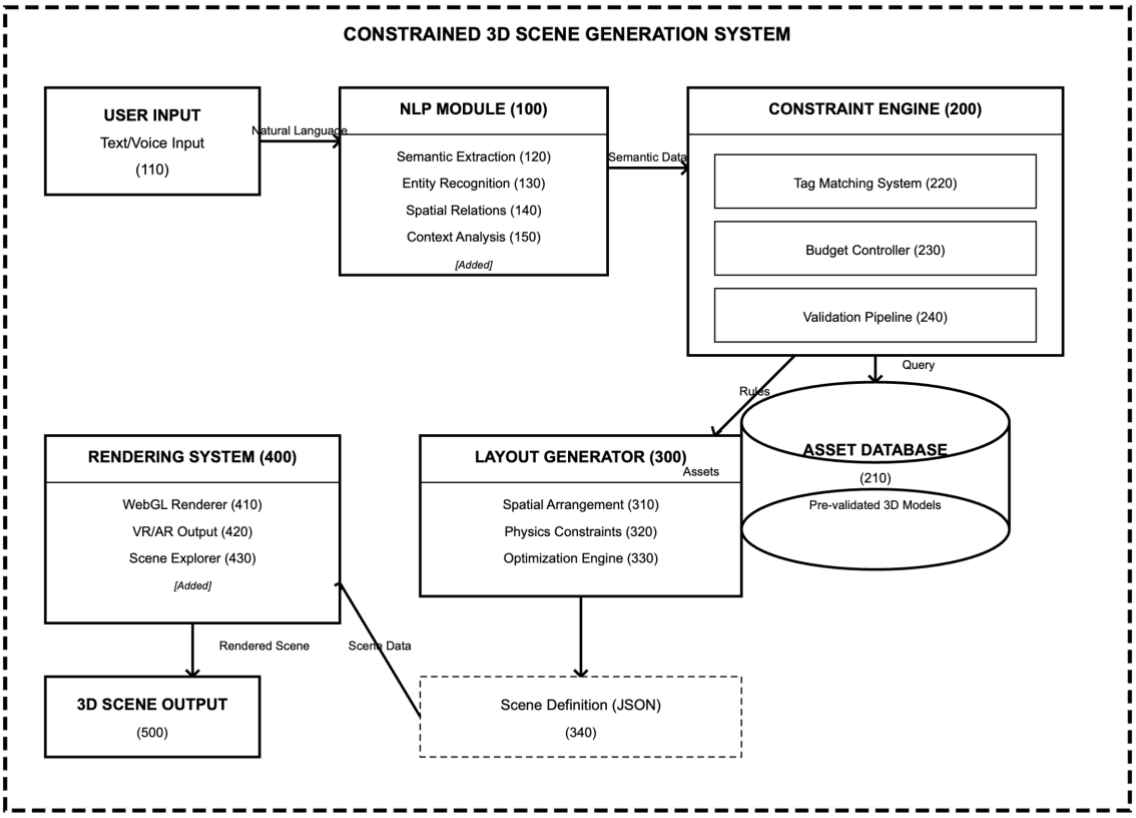


Figure 2: Process Flow Diagram

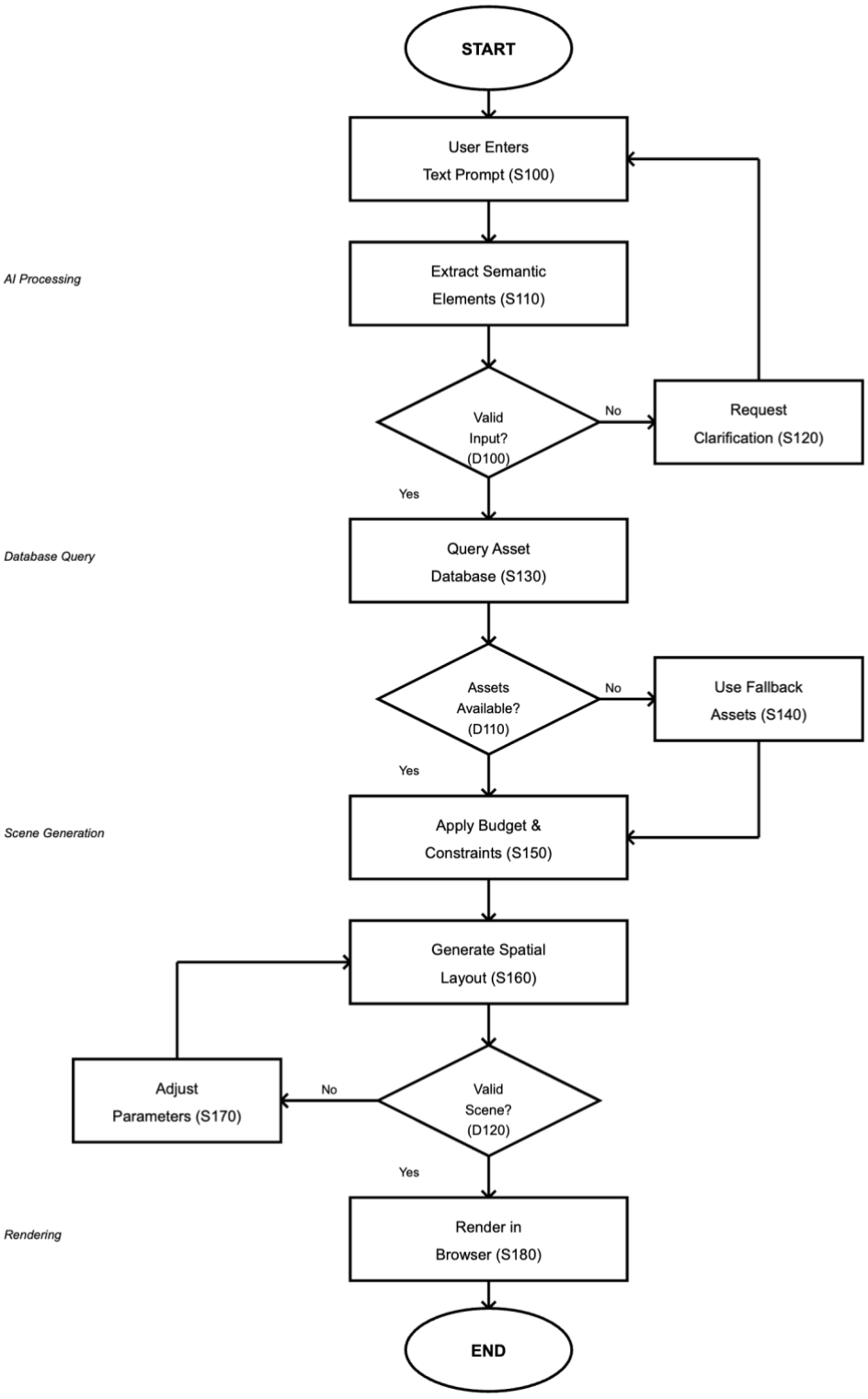
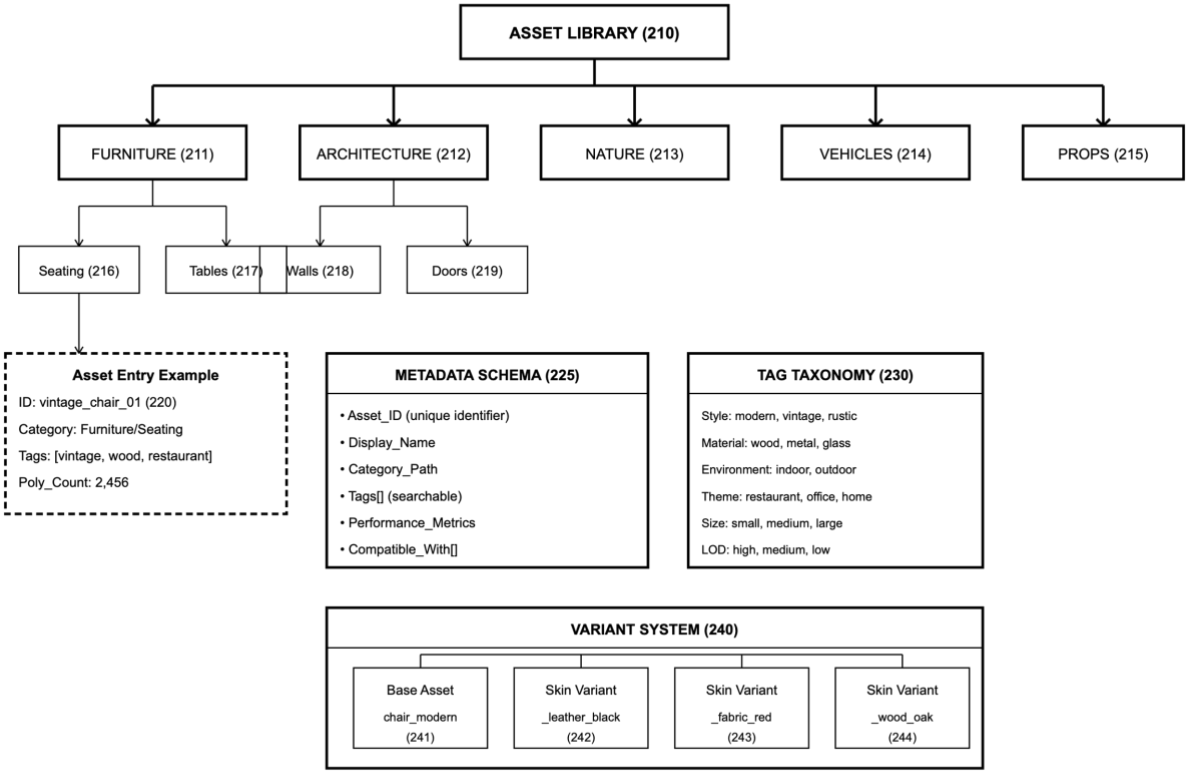


Figure 3: Constrained Asset Library Structure



Note: All assets pre-validated for real-time rendering performance

Figure 4: Natural Language to Scene Transformation

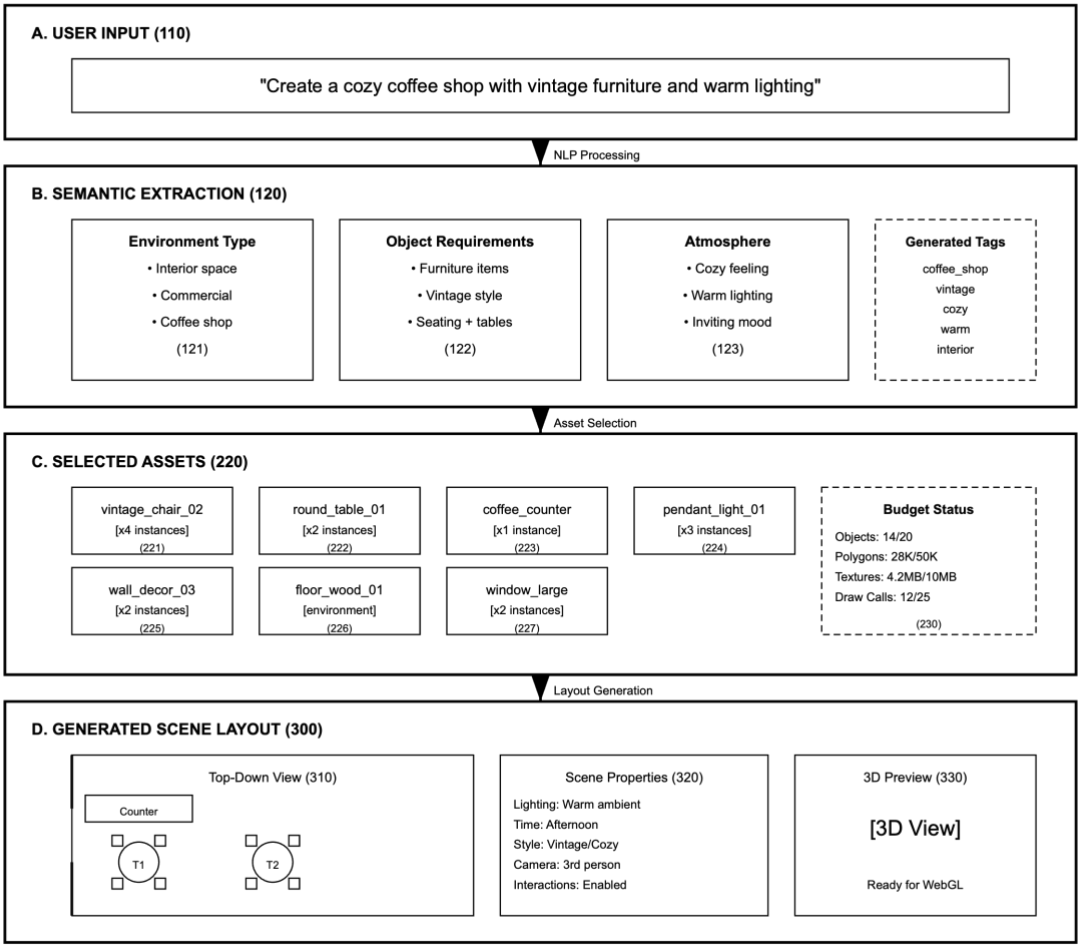


Figure 5: Constraint Engine Decision Tree

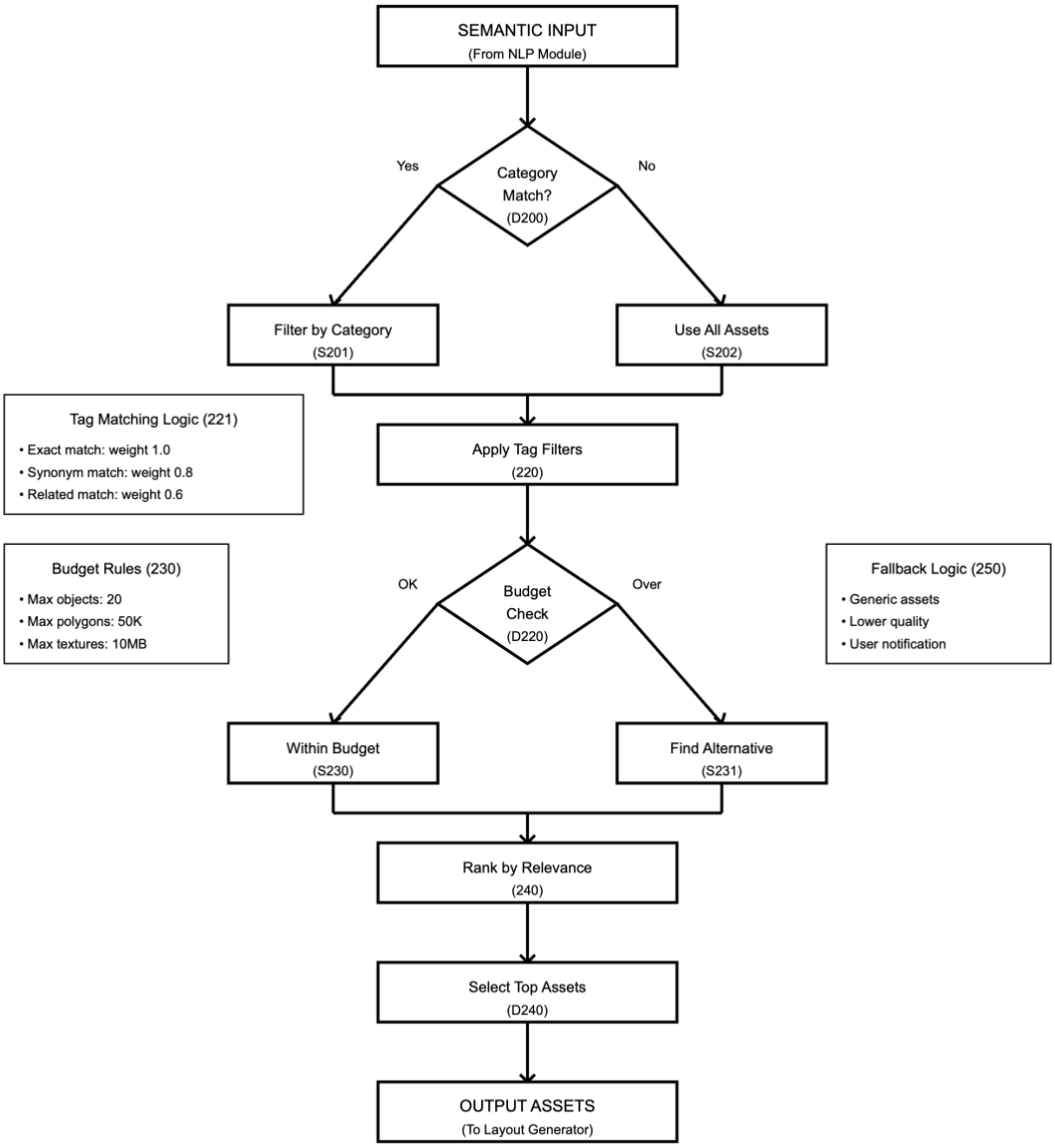
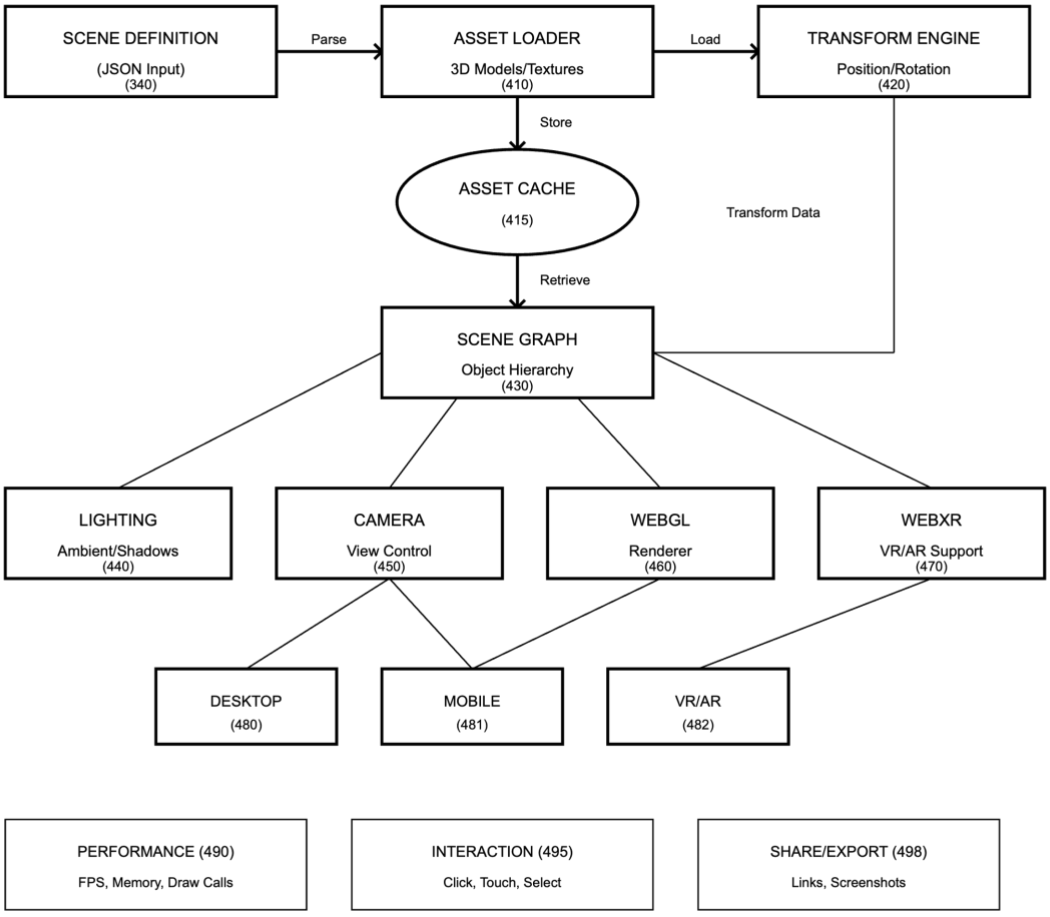


Figure 6: Runtime Rendering Pipeline



All components optimized for 60fps real-time rendering