Constructive Solid Geometry using Signed Distance Fields

Shen-Hsiang Houng
National Tsing Hua University
Taiwan

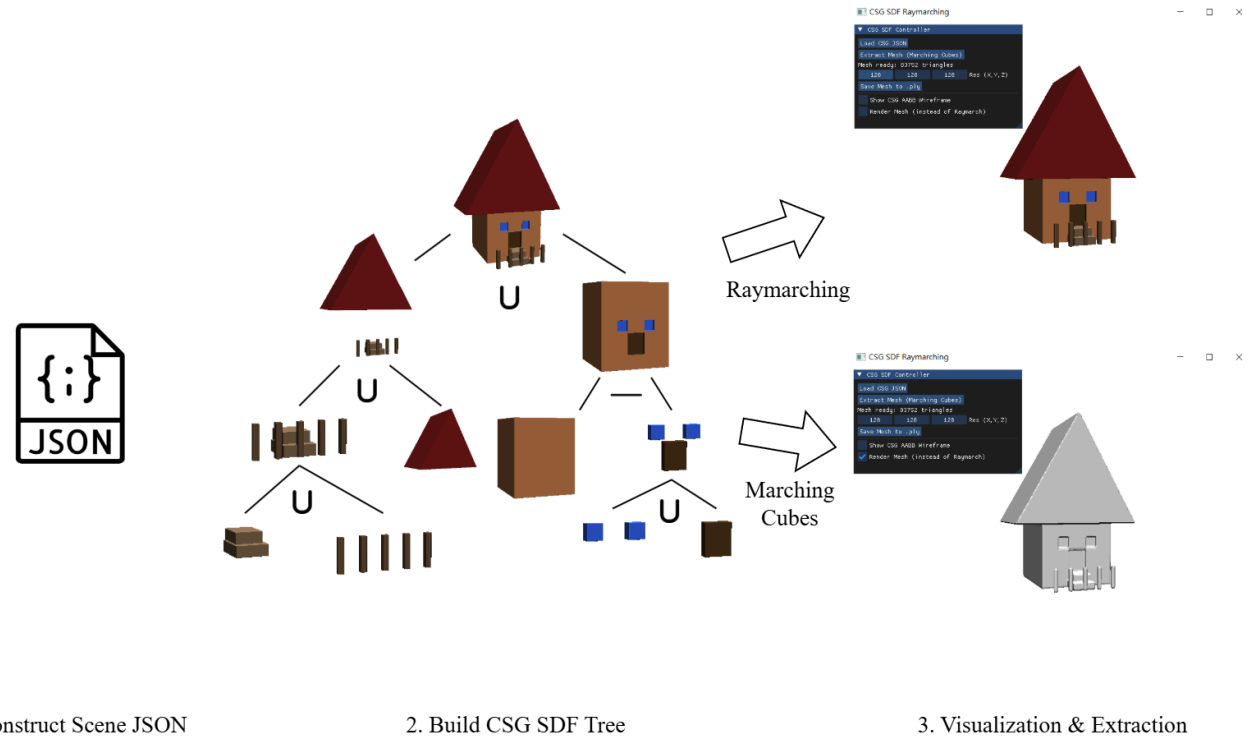


Figure 1: The system parses a scene from JSON, constructs a CSG tree of primitives and operations, performs real-time ray marching to render signed distance fields, and optionally extracts mesh representations via Marching Cubes, with interactive control through a GUI.

1 ABSTRACT

I present a real-time rendering and modeling system based on Constructive Solid Geometry (CSG) using Signed Distance Fields (SDF). The system allows users to define complex 3D shapes by hierarchically combining simple primitives through Boolean operations, specified via a JSON scene description. The CSG tree is traversed on the GPU using efficient ray marching to generate high-quality renderings of implicit surfaces, supporting precise geometry evaluation. To enable interoperability and further processing, we also support mesh extraction via the Marching Cubes algorithm. Additionally, an interactive GUI is provided to inspect the scene, toggle rendering modes, and visualize structure-aware components such as bounding volumes. Our system demonstrates both visual quality and structural clarity in representing complex geometries like houses and robots using only a small set of primitives and operations. To further enhance usability, we integrate a language model interface that allows users to generate custom 3D assets by providing natural language prompts, which are translated into valid scene descriptions. The system demonstrates robust shape composition, efficient rendering, and a novel integration of LLM-assisted authoring for intuitive asset creation. The source code is available at **https://github.com/Sunnyhong0326/SDFBooleanEngine.**

2 INTRODUCTION

Constructive Solid Geometry (CSG) [3] is a powerful technique for modeling complex 3D shapes by hierarchically combining simple geometric primitives using Boolean operations such as union, intersection, and difference. While widely used in CAD and solid modeling, traditional CSG systems based on boundary representations often suffer from limitations: they are complex to implement, prone to numerical and topological errors, and difficult for non-expert users to manipulate—especially when handling surface intersections or structural changes.

This project proposes an intuitive and robust alternative: a CSG modeling system built on Signed Distance Fields (SDFs). SDFs encode geometry as continuous scalar fields where each point in space represents the signed distance to the nearest surface. This

representation allows Boolean operations to be defined with simple mathematical expressions, naturally resolving surface intersections and supporting smooth blending between shapes. By leveraging SDFs, the system avoids the common pitfalls of mesh-based CSG, offering a more stable and expressive modeling foundation.

To enhance interactivity, the system supports real-time rendering through ray marching [1], enabling users to receive immediate visual feedback as they build and modify scenes. Additionally, the system implements the Marching Cubes algorithm [2] for converting implicit SDF representations into polygonal meshes, which can then be exported to standard formats suitable for 3D printing, game engines, or digital content creation.

To further lower the barrier to entry, the system introduces language-model-assisted scene generation. By leveraging large language models (LLMs), users can describe desired shapes in natural language—such as "a house with a slanted roof and two square windows"—and receive a corresponding JSON-based CSG scene. This feature democratizes asset creation by bridging the gap between high-level intent and low-level representation, enabling non-experts to create complex solid geometry without manual scripting.

The primary goal of this project is to explore efficient and expressive geometric modeling techniques while designing a user-friendly interface that empowers users to construct complex solid objects with ease.

3 METHODOLOGY

The overall framework is illustrated in Figure 1. Our system is composed of five modular and extensible components that support efficient modeling and interactive exploration of implicit geometry. First, the scene is parsed from a JSON file and a CSG tree is constructed, as described in Section 3.1. Signed Distance Field (SDF) values are then evaluated by traversing this tree structure (Section 3.2). For real-time feedback, we employ GPU-based ray marching to render the implicit surfaces (Section 3.3). If a mesh representation is needed, our system supports polygonal extraction using the Marching Cubes algorithm (Section 3.4). Finally, to enhance usability, we integrate a language model interface that allows users to define scenes through natural language prompts, which are converted into valid CSG JSON descriptions (Section 3.5).

3.1 CSG tree construction from JSON

Scenes are defined in a structured JSON format that encodes primitive nodes (e.g., boxes, spheres, prisms) and Boolean operation nodes (union, intersection, subtraction). Each node contains type identifiers, transformation parameters, and material properties (e.g., color). During parsing, the system constructs a CSG tree, where internal nodes represent operations and leaf nodes correspond to geometric primitives.

3.2 SDF evaluation via tree traversal

Each primitive defines a corresponding Signed Distance Function (SDF) that computes the shortest distance from a 3D query point to its surface. All supported primitives are shown in Table 1.

3.3 Real-time Visualization Using GPU Ray Marching

The system employs ray marching on the GPU to render the implicit surfaces defined by the SDF. Rays are cast per pixel, and step sizes are dynamically determined by the current SDF value. When the distance to the surface drops below a small threshold (ε), the surface is considered hit, and shading is applied.

The surface normal at point p is estimated using a tetrahedral central difference scheme, evaluating the signed distance function $f(p)$ at four offset directions. The gradient is normalized to yield the unit surface normal.:

$$\mathbf{n}(p) = \frac{\mathbf{k}_{xyy} \cdot f(p + h \cdot \mathbf{k}_{xyy}) + \mathbf{k}_{yyx} \cdot f(p + h \cdot \mathbf{k}_{yyx}) + \mathbf{k}_{yxy} \cdot f(p + h \cdot \mathbf{k}_{yxy})}{\left\| \mathbf{k}_{xyy} \cdot f(p + h \cdot \mathbf{k}_{xyy}) + \mathbf{k}_{yyx} \cdot f(p + h \cdot \mathbf{k}_{yyx}) + \mathbf{k}_{yxy} \cdot f(p + h \cdot \mathbf{k}_{yxy}) \right\|}$$

where $f(p)$ is the signed distance function and the kernel vectors are defined as:

$$\mathbf{k}_{xyy} = (1, -1, -1), \quad \mathbf{k}_{yyx} = (-1, -1, 1)$$

$$\mathbf{k}_{yxy} = (-1, 1, -1), \quad \mathbf{k}_{xxx} = (1, 1, 1)$$

3.4 Mesh extraction via Marching Cubes

To allow export and downstream use of the created models, the system optionally supports polygonization of the implicit surface using the Marching Cubes algorithm. A uniform grid is sampled across a user-defined bounding volume, and the SDF is evaluated at each grid point.

For each voxel cube, a case index is generated based on the sign of its corner SDF values, and triangles are emitted using a precomputed lookup table. The resulting mesh is exported and can be rendered directly in the system or saved for external use.

3.5 Natural Language-driven scene generation using LLMs

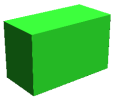
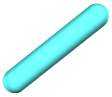
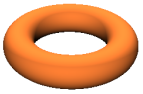
To make scene creation more accessible, the system integrates a natural language interface powered by a large language model (LLM). Users can describe desired scenes (e.g., "a snowman with a hat and buttons") in plain English, and the model generates a corresponding JSON representation that defines the CSG tree.

To achieve this, a prompt template encodes the scene grammar, available primitive types, and composition rules. The generated JSON is then validated and loaded into the system for immediate visualization. This feature bridges the gap between high-level design intent and low-level scene specification, enabling non-experts to rapidly prototype 3D models.

4 RESULTS AND EXPERIMENTS

We evaluate the proposed system by examining (1) the ability of a large language model (LLM) to generate structured 3D scenes through natural language prompts, and (2) a comparison between ray-marched visualizations and mesh extraction results for the generated scenes.

Table 1: Signed Distance Functions (SDFs) for supported primitives with aligned illustrations

Primitive	SDF Equation	Illustration
Box	$\text{sdBox}(\mathbf{p}, \mathbf{b}) = \lVert \max(\lvert \mathbf{p} - \mathbf{c} \rvert - \mathbf{b},\ \mathbf{0}) \rVert$	
Sphere	$\text{sdSphere}(\mathbf{p}, r) = \lVert \mathbf{p} - \mathbf{c} \rVert - r$	
Triangular Prism	$\text{sdTriPrism}(\mathbf{p}) = \max(\text{sdTri2D}(\mathbf{p}_{xz}),\ \lvert y \rvert - h)$	
Cone	$\mathbf{q} = \left(\sqrt{x^2 + z^2},\ y\right)$ $\mathbf{c} = (r,\ -h)$ $\text{sdCone}(\mathbf{p}) = \max\left(\frac{\mathbf{q} \cdot \mathbf{c}}{\lVert \mathbf{c} \rVert},\ -y\right)$	
Cylinder	$\text{sdCylinder}(\mathbf{p}) = \max\left(\sqrt{x^2 + z^2} - r,\ \lvert y \rvert - \frac{h}{2}\right)$	
Capsule	$\text{sdCapsule}(\mathbf{p}) = \left\lVert \mathbf{p} - \mathbf{a} - \frac{(\mathbf{b}-\mathbf{a}) \cdot (\mathbf{p}-\mathbf{a})}{\lVert \mathbf{b}-\mathbf{a} \rVert^2} (\mathbf{b} - \mathbf{a}) \right\rVert - r$	
Torus	$\text{sdTorus}(\mathbf{p}, R, r) = \sqrt{(\lVert \mathbf{p}_{xz} \rVert - R)^2 + y^2} - r$	

4.1 LLM-Prompted Scene Generation

To assess the usability and expressiveness of our language-model interface, we supplied a series of natural language prompts describing simple 3D structures. The LLM was tasked with converting each prompt into a valid JSON-based CSG scene representation using our predefined schema. Table 2 illustrates several examples, including prompts such as:

- *"A robot with a square body, two legs"*
- *"A snowman with three stacked spheres, red round nose and a top black hat"*
- *"A house with a slanted roof, stairs, two square windows and fences in the front"*

The generated JSON trees were syntactically valid and semantically faithful to the prompts. The system was able to produce well-structured CSG hierarchies using the correct primitives (e.g., boxes, spheres, cones) and logical Boolean operations (e.g., unions and subtractions). These results demonstrate the potential of using natural language for intuitive 3D asset creation, significantly lowering the barrier for scene authoring.

4.2 Ray Marching vs. Mesh Extraction

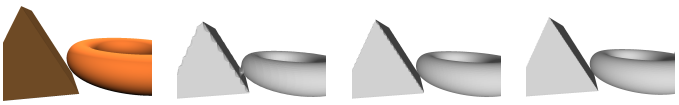
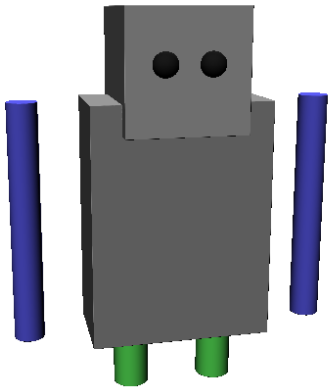


Figure 2: Comparison of ray-marched rendering with Marching Cubes mesh extraction at increasing grid resolutions. Higher resolution improves surface fidelity but increases mesh complexity.

Table 2: Ray-marched results generated from LLM-prompted scenes. Each caption corresponds to the original user prompt.



"A robot with a square body, two legs"

"A snowman with three stacked spheres, red round nose and a top black hat"

"A house with a slanted roof, stairs, two square windows and fences in the front"

To evaluate the visual fidelity of our SDF-based rendering pipeline, we compare real-time ray-marched outputs against mesh reconstructions generated via the Marching Cubes algorithm at varying grid resolutions. Figure 2 shows side-by-side results for representative scenes rendered using ray marching (left), and mesh outputs extracted at low, medium, and high grid resolutions (right).

Ray marching preserves surface smoothness, sharp silhouette boundaries, and fine geometric features due to its continuous evaluation of the underlying SDF. In contrast, Marching Cubes introduces resolution-dependent artifacts such as blocky surfaces, aliasing, and faceting, especially in low-resolution grids. As resolution increases, mesh quality improves, but at the cost of increased vertex count and processing time.

This comparison highlights the trade-off between real-time, high-fidelity rendering using SDFs and the compatibility of polygonal meshes with traditional rendering pipelines. Our system allows users to choose the appropriate representation based on visual needs and downstream use cases.

5 CONCLUSION

In this project, we presented an interactive CSG modeling system based on Signed Distance Fields (SDFs), featuring real-time rendering via GPU ray marching, mesh extraction via Marching Cubes, and a novel natural language interface for scene creation. By representing geometry as SDFs and structuring scenes through CSG trees, our system enables robust Boolean operations, smooth surface blending, and flexible shape composition without concerns of mesh intersection or topological artifacts.

The integration of a large language model further enhances accessibility by allowing users to define 3D scenes through intuitive natural language prompts, which are automatically translated into structured JSON representations. Our experiments demonstrate that the generated scenes are both structurally valid and visually coherent, offering a compelling bridge between high-level design intent and low-level geometry construction.

Through comparisons between ray-marched renderings and polygonal mesh extractions, we highlight the trade-offs between visual fidelity and downstream compatibility. While ray marching captures the smoothness and continuity of SDF-based geometry, Marching Cubes enables interoperability with external tools and formats.

Overall, our system offers a user-friendly, extensible pipeline for procedural 3D modeling and visualization, and opens up new possibilities for accessible shape authoring and educational applications. Future work may explore the incorporation of optimization-based shape fitting, hierarchical level-of-detail representations, or interactive editing directly within the language-guided workflow.

REFERENCES

[1] John Hart. 1995. Sphere Tracing: A Geometric Method for the Antialiased Ray Tracing of Implicit Surfaces. The Visual Computer 12 (06 1995). https://doi.org/10.1007/s003710050084

[2] William E. Lorensen and Harvey E. Cline. 1987. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. In Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH). ACM, 163–169. https://doi.org/10.1145/37401.37422

[3] A.A.G. Requicha and H.B. Voelcker. 1977. Constructive Solid Geometry. Production Automation Project, College of Engineering and Applied Science, University of Rochester. https://books.google.com.tw/books?id=kaAxnQEACAAJ