

SDFSspace

Editing Meshes Through Distance Fields with Coding Agents

Lee Penkman¹, GPT², Claude³

¹Independent researcher ²OpenAI gpt-6.1-sol, run through the codex-infinity harness ³Anthropic Claude Sonnet 5.5

Code, assets and live demo: github.com/lee101/sdfs-space sdfs-space.app.nz

Abstract. We present SDFSspace, a pipeline that turns any triangle mesh into an editable signed distance field, lets a coding agent change the shape by writing a few lines of GLSL, and returns a textured, game-ready mesh. The field is a sparse brick grid that a GPU evaluates directly; building one takes seconds to about a minute, and the extracted mesh stays within 0.033% of the bounding-box diagonal of the source (95th percentile, median over 10 assets). On the textured dragon the full round trip reproduces the source renders at 40 to 49 dB PSNR. Edits are produced by a goal-driven loop in which GPT writes the edit, renders annotated before/after previews, looks at them and revises, while Claude reviews the result independently and returns feedback until the request is met. Across 11 edit requests, 8 are accepted by the reviewer with score 8 or higher, and on the 11 requests we also ran as a single blind pass the loop raises the mean reviewer score from 4.7 to 7.7 (0 of 11 pass in a single blind pass), with 5 accepted in the first round.

1 Introduction

Triangle meshes are the format that engines, printers and marketplaces consume, but they are awkward to edit with code. A distance field is the opposite: adding a horn is a capsule and a smooth union, carving a window is a subtraction, and the same field renders directly on a GPU. Coding agents are good at exactly this kind of short geometric program, which makes distance fields an attractive working representation if the trip into the field and back is faithful enough that nothing is lost on the way.

SDFSspace makes that trip cheap and accurate. A mesh is converted into a sparse brick distance field (Section 3). The agent edits it through a small GLSL layer that composes with the field (Section 4). The edited field is extracted back to a decimated, UV-unwrapped mesh whose PBR maps are transferred from the source textures, so edits keep the material of the original and can carry their own (Section 5). Everything runs on one consumer GPU: in the dragon example the field is built in 65 s, the mesh extracted in 15 s, and the textured export finishes in about two minutes.

Our contributions:

- A brick distance field built directly from meshes, evaluated by GLSL on the GPU, with a conservative far field, automatic handling of open surfaces, and a persistent edit layer (Section 3).
- A fast round trip back to a textured mesh: GPU sparse extraction, parallel atlas packing, and projection of source albedo, roughness, metallic, ambient occlusion and object-space normals onto the new surface (Section 5).
- A goal-driven agent loop with annotated previews, probe and ray tools, and an independent reviewer, evaluated on 11 requests and compared with a single blind pass (Section 6, Section 7.3).
- An open-source implementation, reproducible benchmark suite and live site.

2 Related Work

Sparse volumes are the standard way to store distance fields at high resolution: adaptively sampled distance fields [1], OpenVDB [2] and its GPU form NanoVDB [3]. Our bricks follow this lineage with 8^3 cells, one-cell apron for seamless trilinear lookup, and a conservative coarse level for rays that are far from the surface. Signs for open or imperfect meshes come from fast winding numbers [4], and rays march with sphere tracing [5]. Surface extraction uses surface nets [6] followed by Newton projection onto the field, quadric-style decimation [7] through meshoptimizer [8], and xatlas [9] for charting.

Signed distance functions are also the language of procedural modelling and shader art [10], and neural variants learn them from data [11]. We instead keep the field explicit so that edits are ordinary code. LLM-driven procedural

3D modelling [12] and agent loops that act, observe and revise [13], [14] motivate our edit loop; using a second model as the judge follows common practice for open-ended outputs [15].

3 Brick Distance Fields

Given a mesh M , we choose a voxel size h from the mean edge length, $h = \text{clip}(0.4|e|, \frac{D}{1500}, \frac{D}{500})$ with D the bounding-box diagonal, and cover space with bricks of 8^3 cells. A brick is active when the signed distance at its center is within the brick’s circumradius plus $1.5h$ of zero. Active bricks store 9^3 samples so that trilinear interpolation never leaves the brick. Signed distances use libigl’s closest-point query with fast winding numbers for the sign, evaluated in parallel on the CPU. Samples are stored as half floats; the index is one integer per brick and a half-float coarse level supplies a conservative bound $|d| - \sqrt{3}\frac{s}{2}$ for inactive bricks, with $s = 8h$. Outside the grid, the field adds the distance to the box. The generated GLSL function `fittedSdf(p)` is a few dozen lines and reads three storage buffers.

When more than 10% of probe points just off the surface have ambiguous winding numbers (open sheets, soup), the builder switches to a shell: the field becomes the unsigned distance minus $1.5h$, which keeps thin open geometry such as the beetle’s wings as a closed thin solid.

The field is exact where it matters. Against signed distances computed on the mesh, the dragon field at $h = 0.006$ (0.07% of the diagonal) has a median error of $1.9\text{e-}5$ and a 95th percentile of $2.4\text{e-}4$ world units within 0.02 units of the surface, against a dragon that is 8.6 units across. CPU and GPU evaluations agree to 8×10^{-7} .

4 Editing Layer

An edit is a GLSL file composed with the base field:

```
vec3 editPosition(vec3 p);           // domain warp
float editDistance(vec3 p, float d); // d is the base field
vec4 editColor(vec3 p);              // rgb, alpha = 1 on new geometry
void editMaterial(vec3 p, inout float roughness, inout float metalness);
```

The final field is `editDistance(p, fittedSdf(editPosition(p)))`. Helpers cover spheres, capsules and smooth unions with a polynomial blend; agents add tapered cones, ellipsoids, bends and twists as needed. `editColor` and `editMaterial` give new geometry its own albedo, roughness and metalness: the crown example paints a gold, metallic band with red jewels. The base field is never modified, so every edit is a small readable diff against the original.

5 Back to a Textured Mesh

Extraction. A sparse 8^3 -brick grid at cell size $0.85h$ samples the field on the GPU (about 94 million corners for the dragon in under a second), surface nets produce a mesh, and vertices are Newton-projected onto the zero set using the field gradient. `meshoptimizer` simplifies to the target triangle count with smoothed normals as attributes.

Unwrapping. `xatlas` is single-threaded within one mesh and dominated the original pipeline (328 s for 500k triangles). We split the simplified mesh into 64 spatial parts by recursive median cuts on triangle centroids and let `xatlas` chart the parts concurrently; charts are packed into one atlas. The extraction stage drops from 328 s to 32 s with the same atlas utilization (72%).

Baking and texture transfer. Each covered texel is rasterized on the CPU and its position, frame and footprint are passed to a GPU kernel, which casts a ray from a cage onto the full field and evaluates the adapter’s material. For meshes with source textures, the bake additionally records each texel’s position; we project it to the closest point on the source triangle mesh, interpolate the source UV, and bilinearly sample basecolor, roughness, metallic, specular, ambient occlusion, subsurface, thickness and the object-space normal map. The object-space normal is re-expressed in the new tangent frame, which preserves fine relief that is below the field resolution. New geometry therefore inherits the nearest original surface (a lengthened tail keeps scales, a stretched wing keeps its membrane), while texels colored by an edit’s `editColor` keep the baked value of the edit. A 3×3 despeckle pass removes isolated texels where the closest triangle belongs to a neighboring source island, and gutters are dilated.

6 Goal-Driven Editing Agent

The agent receives a request and a workspace containing `edits.glsl`, the model’s coordinates and four tools: preview renders before/after views from the actual GLSL with coordinate rulers every half unit and optional close-ups; probe samples the field; ray returns surface hits along a ray, which is how the agent places features on the body; and a compile check built into the preview. The prompt asks for at least three preview-and-critique cycles and tells the agent to view each image before continuing. When the agent finishes, a reviewer, Claude Sonnet 5.5 acting only on the before/after sheet, returns a score from 1 to 10, whether the request is achieved, whether anything unrelated was damaged, and actionable feedback. If the score is below 8, the feedback and the latest sheet are sent back and the agent continues from its current file; the best-scoring round is kept. The agent is gpt-6.1-sol through the codex-infinity harness [16], with medium reasoning effort for the first three rounds and high effort afterwards; the reviewer never sees the agent’s explanations.

7 Experiments

7.1 Round-trip fidelity

We prepared ten meshes from the common 3D test model collection [17], scaled to the dragon’s size (Table 1). For each, a brick field is built at voxel multiples of 1, 2, 4 and 8 times the default, extracted at the source triangle budget, and compared with the source by two-sided sampled surface distance and by silhouette IoU over six orthographic views.

Field	Size (KB)	Chamfer mean (% diag)	p95 (% diag)	Mean IoU	Build (s)
bricks $\times 1$	11205	0.018	0.033	0.9917	5.3
bricks $\times 2$	2713	0.024	0.054	0.9846	1.5
bricks $\times 4$	696	0.04	0.099	0.9721	0.5
bricks $\times 8$	193	0.093	0.291	0.9491	0.2
spheres+sheets 1024	338	0.132	0.366	0.9119	8.3
spheres+sheets 4096	434	0.11	0.248	0.8997	9.6

Table 1: Median over ten assets (IoU is the mean). Bricks at the default voxel size reach 0.033% p95. At matching size (about 200 to 350 KB) bricks at $\times 8$ and the compact analytic backend of spheres and curved sheets are comparable; the analytic model has named, hand-editable primitives, the brick field keeps improving with size.

7.2 Textured dragon round trip

The dragon [18] has an 8192^2 texture set baked from its procedural shader. We rebuild it at 500k triangles and a 4096^2 atlas through the full pipeline and compare against the source in Blender Cycles with identical lights and

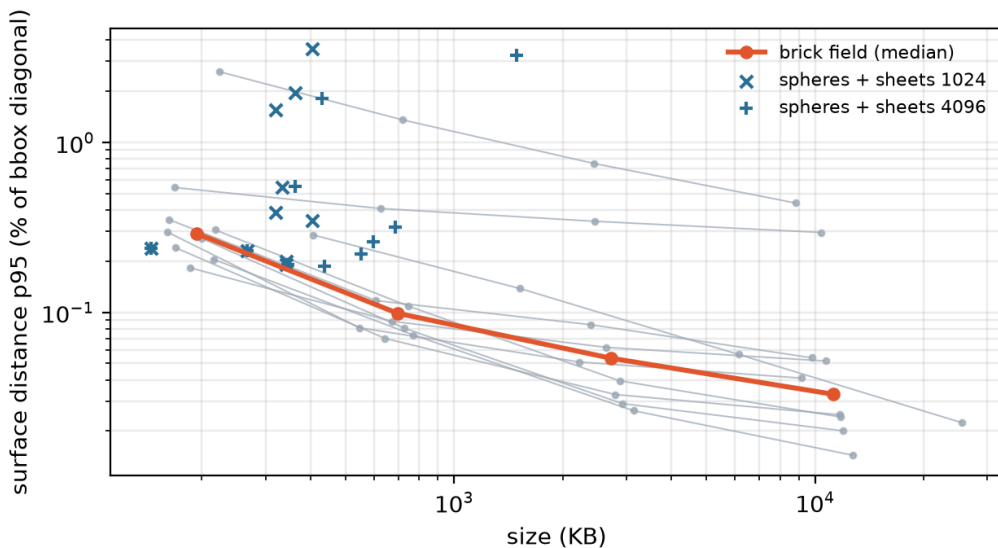


Figure 1: Fidelity against size. Gray lines: individual assets with bricks at four voxel sizes; bold: medians; crosses: the analytic backend. Open surfaces (beetle) are plotted with their shell offset.



Figure 2: Source dragon (left) and the mesh recovered from its brick field with transferred textures (right), rendered in Cycles with identical settings.

cameras (Figure 2). Mean surface distance is 0.00029 world units (0.0033% of the diagonal), p95 0.0095%, and the renders match at 47.4 dB (full view), 48.8 dB (front) and 40.4 dB (face close-up), SSIM above 0.994. The full pipeline from mesh to textured mesh takes 223 s on one RTX 3090 Ti.

7.3 Agent edits

We wrote 11 requests: six on the dragon (scarier, larger wings, longer tail with a spade tip, chubbier body, tusks and collar, a gold crown) and one each on the cow, bunny, monkey, teapot and armadillo. Each request starts from the unedited brick field of its asset and runs the loop of Section 6. A run stops when the reviewer accepts; requests not accepted after three rounds continue, with high reasoning effort, up to eight.

Request	Reviewer scores by round	Best	Rounds	Agent min	One-shot
make the dragon scarier	6, 6, 8	8	3	35.8	6
give the dragon much larger wings with a wider span, keeping the membrane and finger-bone structure	8	8	1	3.9	6
make the tail twice as long, ending in a spade-shaped tip	8	8	1	4.1	4
make the dragon chubby and cuddly: a rounder belly and thicker arms and legs	7, 7, 8	8	3	9.8	3
add a pair of curved tusks rising from the lower jaw and a spiked collar around the neck	7, 3, 6, 4, 4, 7, 6, 6	7	8	51.8	4
give the dragon a five-pointed golden crown sitting between its horns	6, 6, 7, 6, 7, 7, 6, 7	7	8	41.1	4
turn the cow into a unicorn: add a long spiral horn on its forehead	8	8	1	2.6	5
make the ears much taller and add a round fluffy tail	8	8	1	7.2	5
add a tall pointed wizard hat with a wide brim	8	8	1	2.2	7
add a second spout opposite the first and a taller lid knob	4, 4, 7, 6, 7, 6, 5, 8	8	8	116.4	3
give the armadillo a row of tall spikes along its back	6, 7, 7, 6, 7, 7, 7, 7	7	8	50.9	5

Table 2: Reviewer scores (1 to 10) across rounds, the best round kept, agent time, and the score of a single blind pass without tools or retry. A score of 8 or more with no collateral damage counts as accepted.

8 of 11 requests reach the acceptance bar; the mean best score is 7.7 against 4.7 for the blind pass on the same 11 requests. Figure 3 shows the results as rendered meshes. Edits are local by construction: the median fraction of source surface left untouched (within 0.003 units) is 91%.

7.4 Export fidelity of edited meshes

For delivered edits we sample 200k points on the exported triangle mesh and evaluate the edited GLSL field there. Across 11 exports the median 95th-percentile deviation is 0.0092% of the bounding-box diagonal at 400k triangles.

8 Implementation

The brick builder and edit tooling are Python with libigl and NumPy; the GPU stages are C++ with EGL and OpenGL 4.5 compute and run headless; Cycles renders the comparisons. The repository contains the adapter contract for procedural sources (the original Inigo dragon shader is included as an example), the benchmark suite with download scripts, the edit loop with its prompts and reviewer, tests that run without a GPU and with one, this paper’s source, and the site. The agent harness is codex-infinity [16]; any CLI agent that can run commands and view images can drive the loop.

9 Conclusion

Distance fields give coding agents a compact, readable place to change a shape, and sparse bricks make the trip from a mesh into that space and back effectively lossless at game resolution. With previews the agent can see, a reviewer that judges independently, and a loop that keeps the best round, ordinary requests such as longer tails,



Figure 3: Edit results. Dragon rows: source and edited textured meshes (final export, 400k triangles). Other assets: clay renders of the source and of the edited export. Prompts are listed in Table 2.

wizard hats and gold crowns turn into finished textured meshes in a few minutes. We are releasing the code, suite and edit transcripts so that other agents, fields and requests can be dropped into the same harness.

References

- [1] S. F. Frisken, R. N. Perry, A. P. Rockwood, and T. R. Jones, “Adaptively sampled distance fields: a general representation of shape for computer graphics,” in *SIGGRAPH*, 2000.
- [2] K. Museth, “VDB: high-resolution sparse volumes with dynamic topology,” *ACM Transactions on Graphics*, vol. 32, no. 3, 2013.

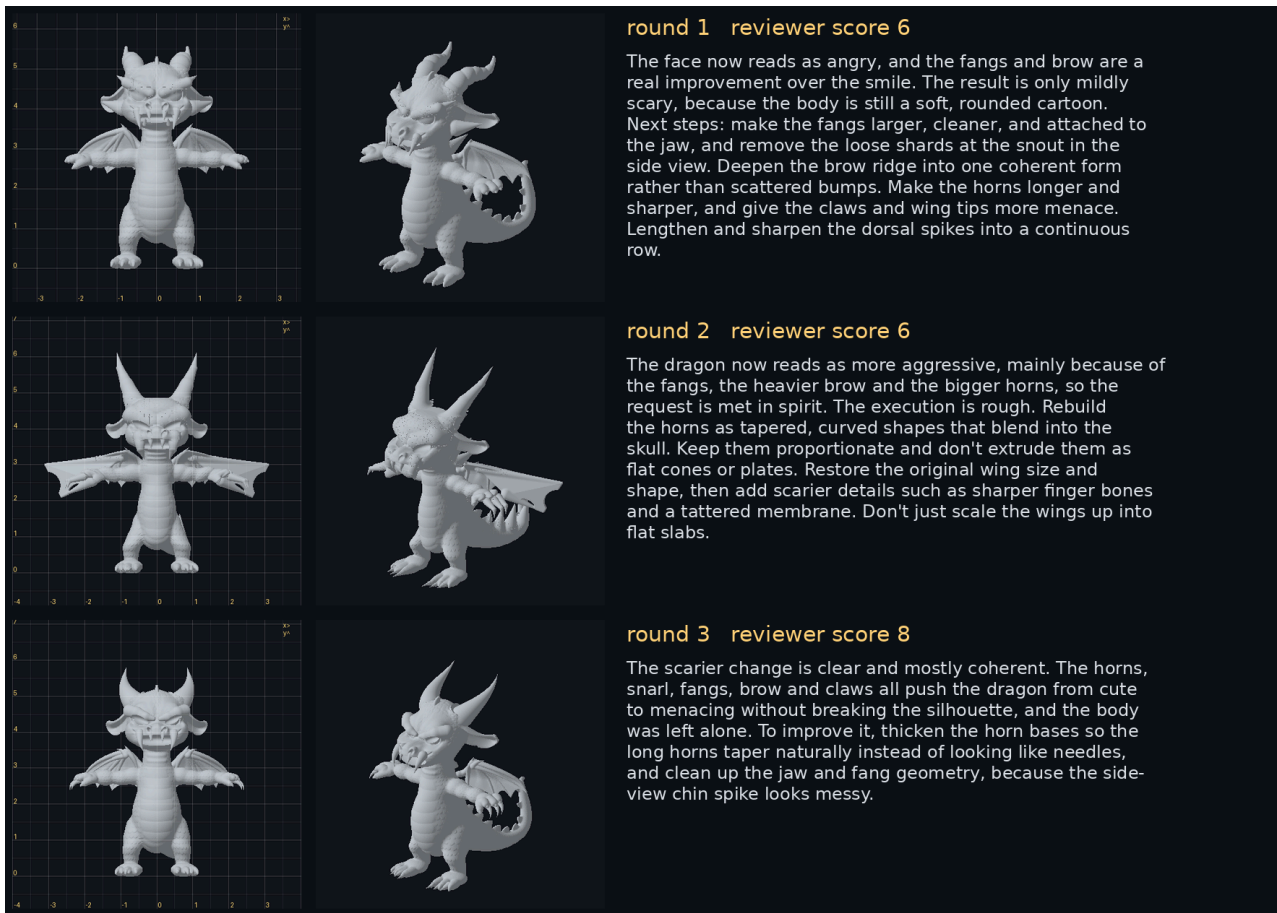


Figure 4: Three rounds of the loop for “make the dragon scarier” (front and three-quarter views) with the reviewer’s critique after each. Round 2 also changed the wings, which the reviewer flags; round 3 restores them, thickens the horn bases and is accepted with score 8.

- [3] K. Museth, “NanoVDB: a GPU-friendly and portable VDB data structure for real-time rendering and simulation,” in *ACM SIGGRAPH Talks*, 2021.
- [4] G. Barill, N. G. Dickson, R. Schmidt, D. I. W. Levin, and A. Jacobson, “Fast winding numbers for soups and clouds,” *ACM Transactions on Graphics*, vol. 37, no. 4, 2018.
- [5] J. C. Hart, “Sphere tracing: a geometric method for the antialiased ray tracing of implicit surfaces,” *The Visual Computer*, vol. 12, no. 10, pp. 527–545, 1996.
- [6] S. Frisken Gibson, “Constrained elastic surface nets: generating smooth surfaces from binary segmented data,” in *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, 1998.
- [7] M. Garland and P. S. Heckbert, “Surface simplification using quadric error metrics,” in *SIGGRAPH*, 1997.
- [8] A. Kapoulkine, “meshoptimizer: mesh optimization library.”
- [9] J. Young, “xatlas: mesh parameterization / UV unwrapping library.”
- [10] I. Quílez, “Distance functions.”
- [11] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove, “DeepSDF: learning continuous signed distance functions for shape representation,” in *CVPR*, 2019.
- [12] C. Sun, J. Han, W. Deng, X. Wang, Z. Qin, and S. Gould, “3D-GPT: procedural 3D modeling with large language models,” *arXiv preprint arXiv:2310.12945*, 2023.
- [13] S. Yao *et al.*, “ReAct: synergizing reasoning and acting in language models,” in *ICLR*, 2023.
- [14] A. Madaan, N. Tandon, P. Gupta, and others, “Self-refine: iterative refinement with self-feedback,” in *NeurIPS*, 2023.

- [15] L. Zheng, W.-L. Chiang, Y. Sheng, and others, “Judging LLM-as-a-judge with MT-Bench and Chatbot Arena,” in *NeurIPS Datasets and Benchmarks*, 2023.
- [16] L. Penkman, “codex-infinity: an infinite coding agent.”
- [17] A. Jacobson, “Common 3D test models.”
- [18] K. Crane, “Inigo the dragon.”