The PhaseTree: Multiphase Signed Distance Fields

ERIC GALIN, Université Lyon 1, INSA Lyon, CNRS, LIRIS, UMR 5205, France
PIERRE HUBERT-BRIERRE, Université Lyon 1, INSA Lyon, CNRS, LIRIS, UMR 5205, France
HUGO SCHOTT, INSA Lyon, Université Lyon 1, CNRS, LIRIS, UMR 5205, France
MARIE-PAULE CANI, LIX, Ecole Polytechnique/CNRS, Institut Polytechnique de Paris, France
ADRIEN PEYTAVIE, Université Lyon 1, INSA Lyon, CNRS, LIRIS, UMR 5205, France
ERIC GUÉRIN, INSA Lyon, Université Lyon 1, CNRS, LIRIS, UMR 5205, France

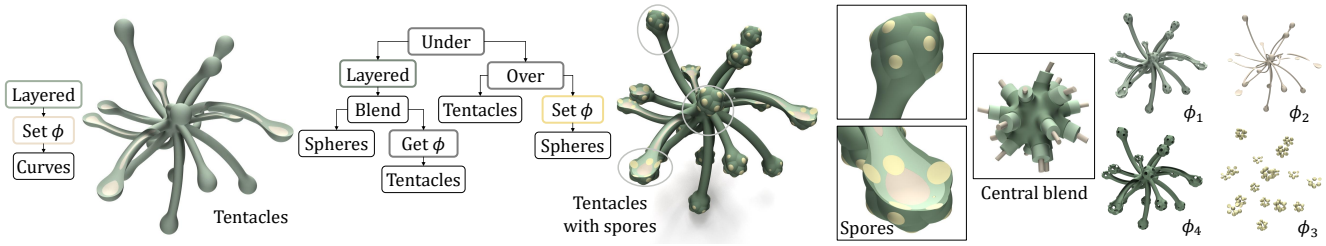

Fig. 1. The PhaseTree is a hierarchical multiphase signed distance representation that supports efficient modeling, fast queries, and real-time rendering of complex volumetric objects with multiple phases, including cross-sections and complex structures. Here, a tentacle model built from a few thousand primitives with four different phases.

We introduce the PhaseTree, a novel hierarchical construction-tree representation for compactly modeling volumetric objects composed of multiple phases or materials across scales. An object is defined as a single construction tree that combines phase-aware primitives through composition and warping operators, yielding a unified multiphase signed distance representation that naturally supports complex topologies and interfaces between phases. The PhaseTree is compatible with standard signed distance field workflows: single-phase algorithms can be directly promoted to a PhaseTree, and conversely reduced without loss of information. As a result, our model integrates seamlessly with existing algorithms and rendering pipelines. We extend classical Sphere Tracing to robustly handle multiphase configurations and show that, despite the additional expressiveness, our implementation preserves the compactness and resolution independence of signed distance fields and incurs less than a 25% runtime overhead compared to single-phase Sphere Tracing.

CCS Concepts: • **Computing methodologies** → **Volumetric models**; **Shape modeling**.

Authors' addresses: Eric Galin, Université Lyon 1, INSA Lyon, CNRS, LIRIS, UMR 5205, France, eric.galin@liris.cnrs.fr; Pierre Hubert-Brierre, Université Lyon 1, INSA Lyon, CNRS, LIRIS, UMR 5205, France, pierre.hubert-brierre@liris.cnrs.fr; Hugo Schott, INSA Lyon, Université Lyon 1, CNRS, LIRIS, UMR 5205, France, hugo.schott@liris.cnrs.fr; Marie-Paule Cani, LIX, Ecole Polytechnique/CNRS, Institut Polytechnique de Paris, France, marie-paule.cani@polytechnique.edu; Adrien Peytavie, Université Lyon 1, INSA Lyon, CNRS, LIRIS, UMR 5205, France, adrien.peytavie@liris.cnrs.fr; Eric Guérin, INSA Lyon, Université Lyon 1, CNRS, LIRIS, UMR 5205, France, eric.guerin@liris.cnrs.fr.

Additional Key Words and Phrases: Implicit surfaces, Signed Distance Fields, Multi-phase objects

ACM Reference Format:
Eric Galin, Pierre Hubert-Brierre, Hugo Schott, Marie-Paule Cani, Adrien Peytavie, and Eric Guérin. 2026. The PhaseTree: Multiphase Signed Distance Fields. *ACM Trans. Graph.* 45, 4, Article 55 (July 2026), 13 pages. https://doi.org/10.1145/3811379

1 INTRODUCTION

Implicit surfaces define the surface as the zero level-set $\partial\Omega = \{\mathbf{p} \in \mathbb{R}^3, f(\mathbf{p}) = 0\}$ of a continuous function f, also referred to as a potential field [Bloomenthal and Wyvill 1997; Wyvill et al. 1986]. Implicit surfaces have been a cornerstone of geometric modeling in Computer Graphics since the early pioneering works of Blinn [Blinn 1982] and Wyvill [Wyvill et al. 1986]. Over the years, their compact formulation, smooth blending operators, warping and suitability for constructive modeling have made them a versatile representation for free-form shapes. Among other implicit representations such as Blobs [Blinn 1982; Wyvill et al. 1986], BlobTree [Wyvill et al. 1999], R-Functions [Pasko et al. 1995], so-called *signed distance fields* have emerged as a widely adopted representation, even if the underlying function f often represents a lower distance bound to the surface.

Hierarchical implicit construction tree models allow for scalable modeling of intricate objects by supporting composition, blending, and deformation in a recursive manner. They enable a variety of accelerated algorithms for efficiently querying the function $f(\mathbf{p})$, such as bounding volumes, levels of detail nodes and accelerated Sphere Tracing algorithms [Hart 1996]. However, while such hierarchical construction tree formulations provide expressive power, they also highlight a key limitation of the standard model, namely modeling heterogeneous structures.

Conventional implicit surfaces inherently describe *monolithic* objects: at any point in space, $f(\mathbf{p})$ encodes only the signed distance to the nearest surface, without differentiating between materials or phases. This mono-phase assumption restricts the representation's ability to model heterogeneous objects, such as composites, layered materials, or volumetric structures with distinct internal phases. In practice, the lack of a consistent and unified way to represent multiple materials or phases within a signed distance field framework forces practitioners to rely on accompanying data structures, multi-channel textures, or auxiliary labeling schemes, none of which integrate seamlessly into existing implicit modeling pipelines.

As an illustrative example, the *offset* operator is well known in traditional mono-phase implicit modeling, where it is used to generate an offset volume from an existing shape. In the richer multiphase setting, this operator naturally generalizes to a new *layered* operator (Figure 1), assigning a distinct phase to the offset region. As depicted in tentacle model in Figure 1, this formulation consistently represents an additional structural layer that encloses or embeds the original shape.

In this paper, we introduce the PhaseTree, a novel hierarchical construction tree representation for modeling *multiphase* implicit surfaces (Figure 1). Our model extends the classical notion of signed distance fields by encoding, within a single consistent data structure, both the signed distance to the closest interface between phases and the local phase at the query point. A key challenge lies in maintaining the consistency and coherence of the individual signed distance fields associated with the different phases when applying modeling operations. The PhaseTree achieves this goal while preserving the simplicity and efficiency of signed distance fields and providing modeling operators for multi-material objects. By embedding phase information in the multiphase distance field itself, we provide a unified foundation for constructive modeling, rendering, and simulation of heterogeneous objects. We demonstrate that our model generalizes classical techniques, integrates naturally with hierarchical implicit trees and accelerated ray-surface intersection algorithms, and opens new opportunities for material-aware modeling.

The main contributions of this work are: 1) A formal definition of multiphase signed distance fields as a generalization of signed distance fields, capable of jointly representing multiple phases or materials; 2) A hierarchical implicit modeling framework enabling constructive modeling of heterogeneous objects compatible with a variety of implicit models such as BlobTrees [Wyvill et al. 1999], function representation [Pasko et al. 1995], and even neural models [Sitzmann et al. 2020]. 3) The preservation of the efficiency and compatibility with existing algorithms such as accelerated Sphere Tracing schemes based on tree pruning or proxies [Barbier et al. 2025; Hubert-Brierre et al. 2025], and Sphere Carving bounding schemes [Schott et al. 2025].

We demonstrate the versatility of our model in various multi-material and layered structures. The explicit identification of phases as consistent signed distance fields enables our model to support a wide range of applications in volumetric modeling. These include the modeling of geological structures, computer-aided design objects, and multi-phase organic shapes, facilitated by the use of warping and smooth blending operators.

2 RELATED WORK

In this section, we focus on implicit modeling for representing complex volumetric models. This work relates to single phase implicit surfaces defining *monolithic* objects, and grid-based distance field representations encoding multiple phases.

We depart from procedural solid textures [Ebert et al. 2002; Tricard et al. 2019; Worley 1996] which are in essence functions $\tau : \mathbb{R}^3 \rightarrow \mathbb{R}^n$ that define the material characteristics $\tau(\mathbf{p})$ at every point in space. Vector solid textures [Wang et al. 2010] characterize regions in space with grid-based signed distance fields to separate distinct regions with their associated solid texture. Volumetric modeling with diffusion surfaces [Takayama et al. 2010], inspired by diffusion curves [Orzan et al. 2008], replaces the solution of a Poisson equation on a voxel grid or tetrahedral mesh with a more efficient algorithm based on cross-sectional diffusion.

While very effective for texturing implicit surfaces by carving shapes in a three dimensional material, solid textures do not provide means to compute the distance to the limit of the material, and therefore do not model the geometry of the interface between different phases. Contrary to standard implicit models textured with a solid texture $\tau(\mathbf{p})$, the PhaseTree model defines the type of material at every point in space $\mathbf{p}$ by computing a 1-Lipschitz signed distance bound to the boundary of the phase.

2.1 Implicit surfaces

The vast variety of existing models come from the various representations of the underlying function f. Grid-based models efficiently compute f by interpolating values stored in an (adaptive) grid [Frisken et al. 2000; Museth et al. 2002, 2005], which, despite some compression schemes, remains memory consuming.

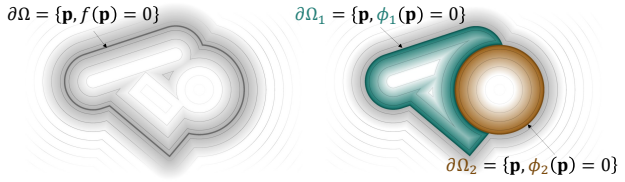

Fig. 2. Comparison of a monolithic implicit surface and a two-phase object.

Procedural models such as Blobs [Blinn 1982; Wyvill et al. 1986] first introduced smooth blending and compact primitive-based representations. The BlobTree model [Wyvill et al. 1999] defined a hierarchical framework supporting warping, blending, and Boolean operations between implicit primitives built from skeletons such as points, line segments, polygons or polyhedra. Pasko *et al.* [1995] presented an alternative function-based representations with a variety of operators. Convolution surfaces [Bloomenthal and Shoemake 1991; Sherstyuk 1999] and variants [Hornus et al. 2003; Zanni et al. 2013] define f as a convolution between a skeleton and a smoothing kernel. While we take inspiration from hierarchical models, contrary to standard mono-phase implicit surfaces, we procedurally define different phases using a multiphase signed distance functions $\Phi : \mathbb{R}^3 \rightarrow \mathbb{R}^n$, which allows us to greatly extend the modeling possibilities (Figure 2).

Recently, neural implicit representations have emerged, exploiting the expressive power of neural networks to encode complex spatial functions [Coiffier and Béthune 2024; Müller et al. 2022; Sitzmann et al. 2020]. These neural fields are typically treated as black-box models and most often approximate the signed distance function only in a narrow band around the surface. They exhibit strong representational power and can capture highly complex geometries, even when learned from rendered images.

In terms of memory efficiency, neural representations are generally less compact than purely functional implicit models, while remaining more memory-efficient than grid-based volumetric representations. While CSG trees may grow significantly for complex models, 1-Lipschitz neural representations are likewise subject to compactness limitations. The size of neural models can be controlled by reducing the number of weights or network parameters, at the cost of diminished geometric detail. In contrast, standard implicit modeling techniques such as instancing and replication enable construction trees to remain relatively compact.

To our knowledge, multiphase neural fields have not yet been proposed with explicit signed distance guarantees and operator closure suitable for constructive modeling, which restricts their applicability to modeling objects with heterogeneous materials or internal phase structure. Nevertheless, our method enables the integration of pre-existing 1-Lipschitz mono-phase neural field as individual phases within a multiphase object.

2.2 Grid-based multiphase models

In the context of level set methods [Zhao et al. 1996], vector-valued implicit functions have been widely used to represent multiphase objects. These formulations were primarily designed for numerical simulation rather than geometric modeling, and are still commonly employed to simulate interacting phenomena such as multiple immiscible fluids [Losasso et al. 2006].

From a modeling perspective, early work on structured volumetric authoring was introduced by Cutler *et al.* [2002], who proposed a tetrahedral mesh-based representation allowing objects to be decomposed into material layers and enabling finite element simulations. More closely related to our approach are multiscale vector volumes [Wang et al. 2011], which combine grid-based signed distance fields within a hierarchical binary tree. This representation resembles a binary space partitioning scheme in which planar separators are replaced by implicit surfaces. Each node stores an signed distance fields sampled on a voxel grid and evaluated using fast tricubic interpolation.

Yuan *et al.* [2012] proposed object-space multiphase implicit functions, where field functions are constructed from labeled sample points defining a decomposition of 3D space into multiple regions. Their representation relies on compact piecewise polynomial functions defined over an adaptively subdivided octree. Despite this hierarchical structure, the method remains memory intensive, typically requiring several megabytes to store a single object.

A common limitation of these approaches is their reliance on grid-based data structures to encode multiple phases. Moreover, they do not provide any guarantees on the signed distance property of the resulting fields; in particular, the generated functions are generally not 1-Lipschitz, which restricts their direct use with standard signed distance field-based algorithms such as Sphere Tracing.

By contrast, our work introduces *multiphase signed distance fields*, a unified extension of classical signed distance functions that integrates phase information directly into a construction tree. This representation preserves the efficiency, compactness, and algorithmic compatibility of signed distance field-based techniques, while providing native support for multi-material and multiphase objects.

3 OVERVIEW

Recall that an implicit surface is defined as: $\partial\Omega = \{\mathbf{p} \in \mathbb{R}^3 \mid f(\mathbf{p}) = 0\}$ where $f : \mathbb{R}^3 \to \mathbb{R}$ denotes a scalar function. In contrast, we define a multiphase object by a multiphase signed distance function $\Phi : \mathbb{R}^3 \to \mathbb{R}^n$ such that $\Phi(\mathbf{p}) = (\phi_1(\mathbf{p}), \cdots, \phi_n(\mathbf{p}))$ (see Figure 2). In our framework, the functions $\phi_i : \mathbb{R}^3 \to \mathbb{R}, i \in [1, n]$ are 1−Lipschitz, although conservative functions, *i.e.*, such that $|\phi_i(\mathbf{p})| \leq d(\mathbf{p}, \partial\Omega_i)$ where d denotes the Euclidean distance and Ω_i the surface of the object of phase i, can also be used.

3.1 Model

The challenge stems from the fact that at any time, a point can only be inside one phase or outside of all phases. Maintaining the consistency between the functions defining the different phases by hand is both time consuming, cumbersome and exhausting for the designer. Our method leverages this demanding step by providing modeling operators that maintain this consistency.

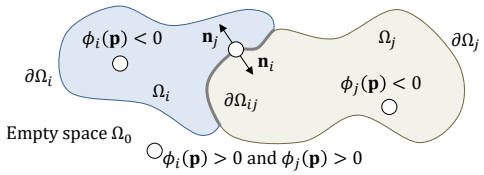

Fig. 3. Notations for multiphase signed distance functions.

Phase functions. ϕ_i must form a partition of space $\mathbb{R}^3$ (Figure 3) into non overlapping phase regions Ω_i to correctly model a multiphase object, this can be expressed as:

$$\Omega_i = \left\{\mathbf{p} \in \mathbb{R}^3 \mid \phi_i(\mathbf{p}) < 0, \text{ and } \phi_j(\mathbf{p}) > 0 \, \forall j \neq i\right\}$$

At each location in space, the function $\Phi(\mathbf{p})$ should have at most one unique negative phase function, indicating the region we are in. Thus, if $\mathbf{p}$ is inside the object, there exists a unique index k such that the corresponding function $\phi_k(\mathbf{p})$ is negative, formally:

$$\forall \mathbf{p} \in \mathbb{R}^3, \quad \exists!\ k \in [1, n] \mid \phi_k(\mathbf{p}) < 0$$

Consequently, the phase type μ at a given point $\mathbf{p}$ is directly defined as the unique index i for which $\phi_i(\mathbf{p}) < 0$, which can be also defined as the one corresponding to the minimum value:

$$\mu(\mathbf{p}) = \arg\min_{i \in [1,n]} \phi_i(\mathbf{p})$$

The surface between the phase k and the other phases is defined as the implicit surface: $\partial\Omega_k = \{\mathbf{p} \in \mathbb{R}^3 \mid \phi_k(\mathbf{p}) = 0\}$. The interface between two phases i and j is the boundary between two phase

regions Ω_i and Ω_j, and the set of points where two phase functions are equal to 0:

$$\partial\Omega_{ij} = \{\mathbf{p} \in \mathbb{R}^3 \,|\, \phi_i(\mathbf{p}) = 0 \text{ and } \phi_j(\mathbf{p}) = 0\}$$

Note that for any point at the boundary between two phases $\partial\Omega_{ij}$, the gradients $\nabla\phi_i$ and $\nabla\phi_j$ are collinear, therefore the normals, defined as normalized gradients $\mathbf{n}_i = \nabla\phi_i/\|\nabla\phi_i\|$ and $\mathbf{n}_j = \nabla\phi_j/\|\nabla\phi_j\|$ respectively, have a consistent opposite direction $\mathbf{n}_i(\mathbf{p}) = -\mathbf{n}_j(\mathbf{p})$ (Figure 3).

By definition, empty regions of space Ω_0 are defined as the points $\mathbf{p}$ that do not belong to any phase $i \in [1, n]$, therefore together the regions Ω_i and Ω_0 form a partition of space:

$$\Omega_0 = \mathbb{R}^3 - \bigcup_{i\in[1,n]} \Omega_i - \bigcup_{(i,j)\in[1,n]^2, i\neq j} \partial\Omega_{ij}$$

They can be defined as points for which every other signed distance function is positive; moreover the signed distance from any point to the outside, *i.e.* to the complementary of the union of all phases, can be consistently defined as ϕ_0 and computed as:

$$\phi_0(\mathbf{p}) = -\min_{i\in[1,n]} \phi_i(\mathbf{p})$$

3.2 Construction tree

The PhaseTree model is defined by a construction tree and takes inspiration from the BlobTree [Wyvill et al. 1999]. The PhaseTree is designed to remain fully compatible with traditional constructive solid geometry and implicit surface workflows. Operators maintain a consistent 1-Lipschitz implicit model for every phase. At the same time, it introduces novel operators such as layering, blending, and multiphase contact, which extend classical monolithic operators to multiphase modeling. Figure 4 shows a typical setup: the leaves of the tree represent monophase signed distance field primitives, whereas the internal nodes represent conversion to phase operators, multiphase combination and warping operators that combine and aggregate their subtrees. Note that the leaves may also contain procedurally defined multiphase primitives, possibly infinite *i.e.* filling space such as the checker primitive (see Section 9.1).

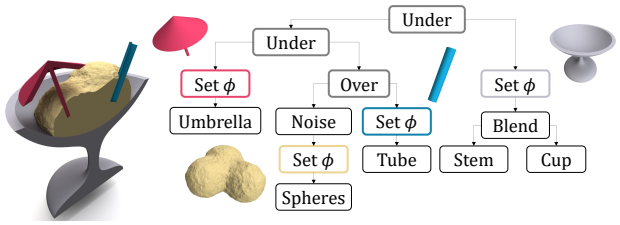

Fig. 4. The ice scoops are generated by adding noise to two sphere primitives. Operators specific to the multiphase model are then used to combine the different phases, with explicit phase priorities in regions where multiple phases overlap. Finally, the umbrella, straw, and glass are assembled using the *under* and *over* operators to produce the complete model.

Figure 5 presents an overview of the different nodes of our model and how it extends signed distance fields. A comprehensive summary of the different operators, including their corresponding equations and cross-sectional illustrations of the associated phases, is provided in Appendix as Figure 28.

The PhaseTree implements a multiphase internal representation, where phases can be retrieved to obtain a signed distance field, whereas any function can be used to define a specific phase ϕ_i. As shown in Figure 5, our model has its own internal phase operators and primitives (see Section 4). In addition, we provide Hybrid operators (see Section 5) that take a signed distance field f as an extra argument, often to define the region of action or region of influence of the operator. Finally, warping that deforms space can be directly applied to both models, or adapted to apply to a restricted subset of phases (see Section 6).

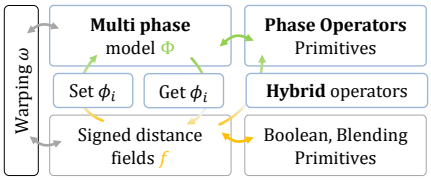

Fig. 5. Overview of the nodes implemented in the PhaseTree model. *Conversion operators* (Set and Get phase ϕ_i) enable bidirectional translation between standard signed distance fields and the multiphase representation. *Phase operators* operate exclusively on multiphase nodes, while *hybrid operators* allow the combination of signed distance fields and multiphase models within a single construction tree. *Warping operators* can be applied to both model types.

Throughout the remainder of this paper, a node will be denoted by a calligraphic symbol such as $\mathcal{A}$, its corresponding phase function $\Phi = (\phi_1, \cdots, \phi_n)$, and the signed distance field of phase k of a node $\mathcal{A}$ will be referred to as $\phi_{\mathcal{A}_k}$.

Unless specified otherwise, the operators introduced for multiphase modeling preserve the 1-Lipschitz property. This condition is essential not only for Sphere Tracing [Hart 1996], but also for robust modeling, which is the primary focus of this work. While conservative fields support Boolean operators, they generally do not accommodate smooth or extended blending operations [Dekkers et al. 2004], motivating our formulation. As shown in [Hubert-Brierre et al. 2025], preserving the 1-Lipschitz property ensures that smooth Boolean operators can be safely applied throughout a signed distance field construction tree. Nevertheless, conservative nodes can still be incorporated in modeling, provided that the nodes above $\mathcal{A}$ consist exclusively of Boolean or warping operators.

4 PHASE OPERATORS

In this section, we describe the different operators, starting with the crucial conversion operator that enables the construction of a PhaseTree from an existing signed distance field. All introduced operators preserve the single-negative-phase invariant and maintain the 1-Lipschitz property, which ensures that smooth Boolean operators can be safely applied anywhere in the tree.

4.1 Conversion from signed distance field models

Signed distance field primitives can directly be used in the PhaseTree model. To convert a standard signed distance field f into a multiphase model Φ, we only need to define it as one of its phases, denoted k (labelled *Set* ϕ in Figures). The corresponding multiphase

model $\mathcal{M}_k(f)$ and its associated function is simply defined as:

$$\Phi = \begin{cases} \phi_k = f \\ \phi_i = \infty \quad \forall i \neq k \end{cases}$$

For all components $i \neq k$, defining $\phi_i(\mathbf{p}) = \infty$ states that the distance between $\mathbf{p}$ and the phase i is infinite, therefore phase i does not exist. In terms of compatibility and performance, it allows to define a complete sub-trees efficiently as a standard signed distance field, provided the function f is 1-Lipschitz. Acceleration nodes such as the proxies or level of detail nodes introduced by Hubert-Brierre *et al.* [2025] can be used in the definition of f without restriction.

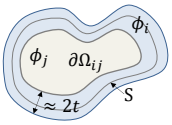

Fig. 6. Thin shell.

Variants of conversion operators exist that optimize computation by reducing the tree complexity. A typical example is the thin shell operator that generates two phases derived from a single function defining the shape of an object. Let f denote the signed distance function of a control shape $\mathcal{S}$. Recall that inflating an object $I(\mathcal{A}, t)$ can be computed as $f - t$; and that the function of the thin shell $T(\mathcal{S})$ can be defined as $|f(\mathbf{p})| - t$, where t is the thickness parameter. The corresponding two-phase object with phase i defining the thin shell (Figure 6) is characterized by:

$$\Phi = \begin{cases} \phi_i = |f(\mathbf{p})| - t \\ \phi_j = f + t \end{cases}$$

4.2 Phase merging

Converting a multiphase model back into a standard signed distance field can be desirable in several situations, for instance to aggregate selected phases or to apply existing algorithms designed for single-phase signed distance fields. To this end, we introduce a *phase merging* operator, denoted M, which fuses a subset of phases into a single signed distance field.

Let $\mathcal{K}$ be the set of indices of the phases to be merged; we define the merged object as:

$$M(\mathcal{A}, i) = \bigcup_{i \in \mathcal{K}} \mathcal{A}_i$$

The associated signed distance field is given by:

$$f = \min_{i \in \mathcal{K}} \phi_{\mathcal{A}_i}$$

Using the merge operator with a single index, i.e. $M(\mathcal{A}, \{k\})$, is equivalent to extracting phase $\phi_{\mathcal{A}_k}$ as an individual signed distance field. More generally, all phases of a multiphase model can be merged into a single signed distance field. Since each $\phi_{\mathcal{A}_i}$ is a 1-Lipschitz function, their pointwise minimum also preserves the 1-Lipschitz property, ensuring that the merged field remains valid. Note, however, that the resulting function satisfies $f(\mathbf{p}) = 0$ at the interfaces $\partial\Omega_{ij}$ between the original phases, as illustrated in Figure 7.

Fig. 7. Remaining zeroes after merging.

4.3 Combination operators

As multiphase operates on a vector of values, Boolean operators such as union, intersection and difference, do not directly adapt to multiphase model. We introduce the following non commutative *under* operator, denoted as $\cup_-$, that performs the union of the phases of its arguments while replacing the phases of first argument by the phases of the second (Figure 8):

$$\mathcal{A} \cup_- \mathcal{B} = \left\{ (\mathcal{A}_i \cup \mathcal{B}_i) - \bigcup_{j \neq i} \mathcal{B}_j \right\}_{i \in [1,n]}$$

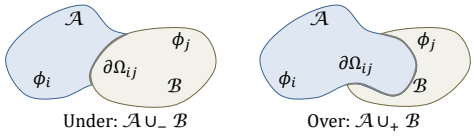

Fig. 8. Under and over operators.

Recall that the union and intersection operators for signed distance fields are defined by using the min and max functions. The corresponding phase functions can be computed as:

$$\forall i \in [1, n], \quad \phi_i = \max(\min(\phi_{\mathcal{A}_i}, \phi_{\mathcal{B}_i}), -\min_{j \neq i}(\phi_{\mathcal{B}_i}))$$

In contrast, the symmetric *over* operator replaces the phases of its second operand with those of the first one; consequently, $\mathcal{A} \cup_+ \mathcal{B}$ is equivalent to $\mathcal{B} \cup_- \mathcal{A}$. Figure 9 demonstrates the expressiveness of these operators by illustrating a rock wall composed of individual blocks assembled using a cement phase.

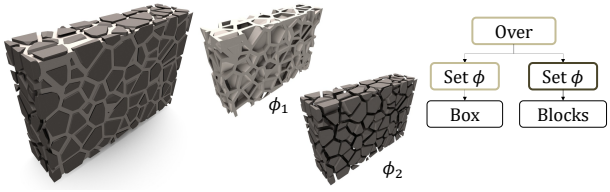

Fig. 9. A rock wall synthesized by combining Voronoi cells modeling individual rock blocks with a box-shaped cement phase, using the *over* operator.

4.4 Contact between phases

Instead of combining two phases using the *over* and *under* operators, it is possible to produce a contact surface between two phases $\mathcal{A}$ and $\mathcal{B}$, inside $\mathcal{A} \cap \mathcal{B}$, as if the objects were colliding. The contact operator takes its inspiration from the work of Gascuel *et al.* [1993], however we do not focus on deformations. It relies on the definition of an implicit surface defined by a signed distance field f.

Let $\phi_{\mathcal{A}_i}$ and $\phi_{\mathcal{B}_j}$ the phase functions of $\mathcal{A}$ and $\mathcal{B}$. We define the 1-Lipschitz signed distance field $f : \mathbb{R}^3 \to \mathbb{R}$ as the weighted sum:

$$f(\mathbf{p}) = \alpha\, \phi_{\mathcal{A}_i}(\mathbf{p}) - (1 - \alpha)\, \phi_{\mathcal{B}_j}(\mathbf{p}) \qquad \alpha \in [0, 1]$$

We denote $\Omega^+ = \{\mathbf{p}, f(\mathbf{p}) > 0\}$ the region of space where $f(\mathbf{p}) > 0$, which is equivalent to $\alpha\, \phi_{\mathcal{A}_i} > (1 - \alpha)\phi_{\mathcal{B}_j}$, and the complementary $\Omega^- = \mathbb{R}^3 - \Omega^+$ respectively. We define the contact operator between two objects with different phases $C(\mathcal{A}, \mathcal{B})$ as:

$$\widetilde{\mathcal{A}} = \mathcal{A} - \Omega^+ \qquad \widetilde{\mathcal{B}} = \mathcal{B} - \Omega^-$$

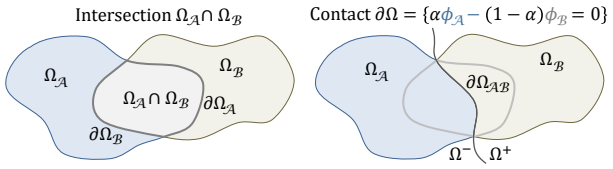

Fig. 10. The implicit surfaces $\partial\Omega_\alpha = \{\mathbf{p}, f(\mathbf{p}) = 0\}$ defines two regions Ω^- and Ω^+ that are used to define the volumes of $\mathcal{A}$ and $\mathcal{B}$ into contact.

The corresponding multiphase function is defined as follows:

$$\Phi = \begin{cases} \phi_i = \max(\phi_{\mathcal{A}_i}, -f) \\ \phi_j = \max(\phi_{\mathcal{B}_j}, f) \\ \phi_k = \infty & \forall k \neq i, j \end{cases}$$

When $\alpha \to 0$, f converges to $\phi_{\mathcal{B}_j}$ and therefore $\Omega^+ \to \Omega_{\mathcal{B}}$, and conversely $\alpha \to 1$, f converges to $\phi_{\mathcal{A}_i}$ and $\Omega^- \to \Omega_{\mathcal{A}}$. Thus, when α varies within the unit interval $[0, 1]$ the contact operator $C(\mathcal{A}, \mathcal{B})$ interpolates the over and under operators $\cup^-(\mathcal{A}, \mathcal{B})$ and $\cup^+(\mathcal{A}, \mathcal{B})$. In particular, when $\alpha = 1/2$, the implicit surface $\partial\Omega = \{\mathbf{p}, f(\mathbf{p}) = 0\}$ separates space into two domains where $\phi_{\mathcal{A}_i} > \phi_{\mathcal{B}_j}$ and $\phi_{\mathcal{A}_i} < \phi_{\mathcal{B}_j}$ respectively (Figure 10).

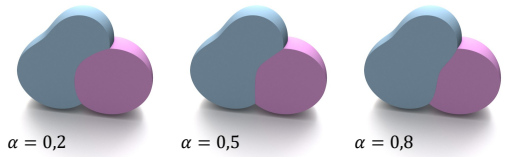

Fig. 11. Contact surface between two phases.

4.5 Layered operator

In practice, it is often convenient to design complex multiphase objects by incrementally layering shapes [Cutler et al. 2002]. The *layered* operator, denoted as L, is an unary operator directly derived from the *under* operator that takes the set of phases $\mathcal{A}_i$, $i \in [1, n]$, and embeds them within a new layer $n + 1$ of thickness t to obtain a new different phase. It generalizes the standard signed distance field offsetting operator to multiphase layers. The rounding operator increases the volume of an existing phase, whereas the layering operator creates a new phase that surrounds and embeds the volume of several phases.

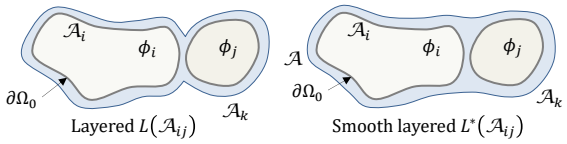

Fig. 12. Example of normal and smooth layered operators creating a new phase $\phi_{\mathcal{A}_k}$ embedding the volume of the two-phase object $\{\phi_{\mathcal{A}_i}, \phi_{\mathcal{A}_j}\}$.

Recall that $\mathcal{A}_0$ is the complementary of the union of all phases, thus $\overline{\mathcal{A}}_0$ denotes the union of existing phases. Formally we have:

$$L(\mathcal{A}) = \mathcal{A} \cup_- \mathcal{A}_{n+1} \quad \mathcal{A}_{n+1} = I(\overline{\mathcal{A}}_0, t) \cap \mathcal{A}_0 \quad \mathcal{A}_0 = \mathbb{R}^3 - \bigcup_{i \in [1,n]} \mathcal{A}_i$$

Therefore, the layered phase function can be computed as follows:

$$\Phi = \begin{cases} \phi_i = \phi_{\mathcal{A}_i} & \forall i \in [1, n] \\ \phi_{n+1} = \max(-\phi_{\mathcal{A}_0} - t, \phi_{\mathcal{A}_0}) & \phi_{\mathcal{A}_0} = -\min\limits_{i \in [1,n]} \phi_{\mathcal{A}_i} \end{cases}$$

This operator can be implemented and optimized to be computationally more efficient when using a sequence of *layered* operators. Note that it is possible to use the smooth union operator instead of the union in the definition of $\mathcal{A}_{n+1}$ to obtain a smooth coating effect between all phases $\mathcal{A}_i$, $i \in [1, n]$. We define:

$$\mathcal{A}_{n+1} = I\Big(\bigcup^*_{i \in [1,n]} \mathcal{A}_i, t\Big) \cap \mathcal{A}_0$$

In this case, the smooth union $\mathcal{A}_i \cup^* \mathcal{A}_j$ of two phases i and j respectively [Dekkers et al. 2004] may be defined as:

$$\phi_{\mathcal{A}_i \cup^* \mathcal{A}_j} = \min\left(\phi_{\mathcal{A}_i}, \phi_{\mathcal{A}_j}\right) - \frac{1}{6} r \left(\max\left(1 - |\phi_{\mathcal{A}_i} - \phi_{\mathcal{A}_j}|/r, 0\right)\right)^3$$

Figure 13, inspired by Cutler *et al.* [2002], illustrates this ability with an entremet composed of a series of layers, the last layer being computed using a smooth union of the underneath cream layers and nuts.

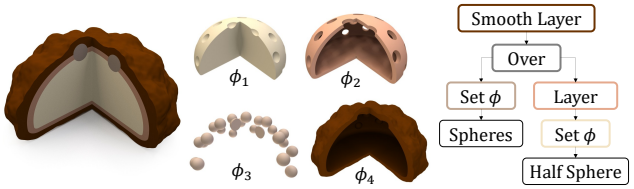

Fig. 13. An entremet modeled through successive phase layers. The outermost layer is obtained by blending the underlying layers using a smooth union operator, resulting in a smooth and continuous covering.

5 HYBRID OPERATORS

Hybrid operators enable phase-aware modifications constrained by an external geometric support, a capability not available in purely multiphase operators.

5.1 Intersection and difference with a shape

It is often practical to compute the intersection, denoted as $\mathcal{B} = \mathcal{A} \cap \mathcal{S}$, of the difference between a multiphase object $\mathcal{A}$ and a shape $\mathcal{S}$ defined by a signed distance field f (Figure 14). We have:

$$\forall i \in [1, n], \quad \mathcal{B}_i = \mathcal{A}_i \cap \mathcal{S}$$

The corresponding phase functions can be computed as:

$$\forall i \in [1, n], \quad \phi_i = \max(\phi_{\mathcal{A}_i}, f)$$

The difference operator is defined similarly by using $-f$ instead.

A powerful variant of the previously defined operator consists in computing the intersection or difference only for a specified set of phase indexes. Such an operator is particularly useful for cutting parts of objects, removing some phases inside a prescribed volume while preserving the others, as demonstrated throughout the paper (see Figure 15 and Figure 17).

Without loss of generality, consider that we only want to compute the intersection with the control shape $\mathcal{S}$ and the phase k, denoted

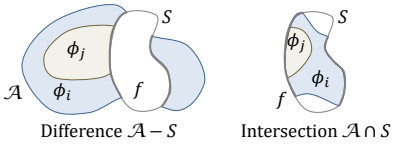

Fig. 14. Intersection and difference between a multiphase object and a shape defined by a signed distance field f.

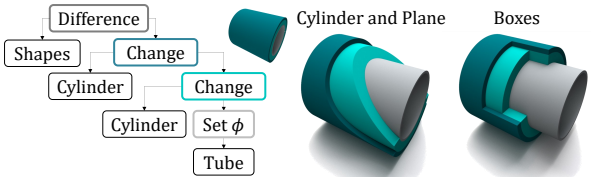

Fig. 15. A reinforced pipe modeled from three coaxial cylinders layers; successive phase-specific difference (cylinder an planes on the left, boxes on the right) enable rich multiphase cross section view and model inspection.

as $\mathcal{B} = \mathcal{A} \cap_k \mathcal{S}$. The equations (Section 4.3) simplify to a less computationally intensive form:

$$\forall i \neq k \quad \mathcal{B}_i = \mathcal{A}_i \qquad \mathcal{B}_k = \mathcal{A}_k \cup \mathcal{S} - \bigcup_{j \neq k} \mathcal{A}_j$$

Thus:

$$\Phi = \begin{cases} \phi_i = \phi_{\mathcal{A}_i} & \forall i \neq k \\ \phi_k = \max\left(\min(\phi_{\mathcal{A}_k}, f), -\min_{j \neq k} \phi_{\mathcal{A}_j}\right) \end{cases}$$

This variant generalizes to any set of phases easily. The second term enforces phase exclusivity by preventing overlaps with competing phases.

5.2 Phase change operators

A practical unary operator is the global phase removal, denoted as $R(\mathcal{A}, k)$, that removes a prescribed phase k of a subtree; it simply preserves all phases and sets ϕ_k to infinity:

$$\Phi = \begin{cases} \phi_k = \infty \\ \phi_i = \phi_{\mathcal{A}_i} \end{cases} \quad \forall i \neq k$$

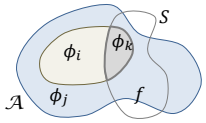

Fig. 16. Local phase change.

Localized phase change, denoted as $C_{i \to k}(\mathcal{A}, \mathcal{S})$, is a powerful local operator that takes as input a multiphase object node $\mathcal{A}$, a region of space $\mathcal{S}$ defined by a signed distance field f, and two phase indices i and k (Figure 16). The operator changes all the phases i into the phase k inside $\mathcal{S}$. Formally, this consists in computing the difference between phase i and $\mathcal{S}$, while completing the volume $\mathcal{S} \cap \mathcal{A}_i$ with phase k:

$$\mathcal{B}_i = \mathcal{A}_i - \mathcal{S} \qquad \mathcal{B}_k = \mathcal{A}_k \cup (\mathcal{S} \cap \mathcal{A}_i)$$

The corresponding signed distance field equations follow:

$$\Phi = \begin{cases} \phi_i = \max(\phi_{\mathcal{A}_i}, -f) \\ \phi_k = \min(\phi_{\mathcal{A}_k}, \max(\phi_i, f)) \end{cases}$$

5.3 Phase-specific smooth-Boolean operators

One of the most distinctive properties of implicit surfaces is their ability to smoothly blend geometries. Recall that, in the context of signed distance fields, smooth-Boolean operators or blending can be obtained by replacing the minimum or maximum of two fields f and g with a formulation that adds or subtracts a compactly supported offset function [Dekkers et al. 2004] (see Appendix 9.3).

Without loss of generality, in the PhaseTree, we define the phase-specific smooth union with shape S over other phases operator $\mathcal{B} = \cup^*_+(\mathcal{A}, S, k)$ with:

$$\mathcal{B}_k = \mathcal{A}_k \cup^* S \quad \text{and} \quad \mathcal{B}_i = \mathcal{A}_i - (\mathcal{A}_k \cup^* S) \quad \forall i \neq k$$

Here $\cup^*$ denotes the smooth union operator with blending radius r. The corresponding phase functions may be computed as:

$$\Phi_{\mathcal{B}} = \begin{cases} \phi_{\mathcal{B}_k} = \widetilde{\phi_k} \\ \phi_{\mathcal{B}_i} = \max(\phi_{\mathcal{A}_i}, -\widetilde{\phi_k}) \end{cases} \quad \forall i \neq k \qquad \widetilde{\phi_k} = s(\phi_{\mathcal{A}_k}, f)$$

with f the signed distance field of the shape S. Smooth intersection and difference, combining with *over* and *under* phase behaviors, can be defined in the same way (see Figure 1).

6 WARPING

The warping operator deforms space, following the classical formulation of space deformation for signed distance fields. Let $\omega : \mathbb{R}^3 \to \mathbb{R}^3$ denote a warping function, we define $\omega(\mathcal{A})$ as:

$$\Phi(\mathbf{p}) = 1/\lambda \; \Phi_{\mathcal{A}} \circ \omega^{-1}(\mathbf{p}) \qquad \lambda \geq \sup \|\mathbf{J}_{\omega^{-1}}\|$$

Here $\Phi_{\mathcal{A}} \circ \omega^{-1}(\mathbf{p}) = (\phi_{\mathcal{A}_1} \circ \omega^{-1}(\mathbf{p}), \cdots, \phi_{\mathcal{A}_n} \circ \omega^{-1}(\mathbf{p}))$ implies that the warping is applied to all phases. Figure 17 illustrates an example of a twisting deformation applied to electrical cables enclosed within a flexible insulating sheath.

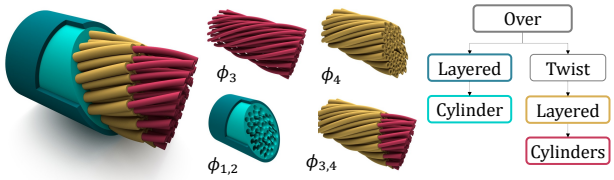

Fig. 17. Twisted electrical cables enclosed within a flexible insulating sheath (simplified tree).

In the multiphase context, warping can be applied to a restricted subset of phases. Without loss of generality, we propose to denote this operator $D(\mathcal{A}, k)$ where k denotes the target deformed phase. It then becomes necessary to define whether the deformed phases should replace or preserve the other phases, using the *over* and *under* operators respectively. This high-level operator may be defined as a combination of previous core operators:

$$D(\mathcal{A}, k) = D(\mathcal{A}_k) \cup_+ \left(\bigcup_{i \neq k} \mathcal{A}_i\right) = D(\mathcal{A}_k) \cup_+ (\mathcal{A} - \mathcal{A}_k)$$

In terms of function evaluation, as we are deforming only one phase k, the equations simplify to:

$$\Phi = \begin{cases} \phi_i = \max(\phi_{\mathcal{A}_i}, -\phi_{\mathcal{A}_i} \circ \omega^{-1}/\lambda) \\ \phi_k = \phi_{\mathcal{A}_k} \circ \omega^{-1}/\lambda \end{cases} \quad \forall i \neq k$$

7 RESULTS AND DISCUSSION

We implemented the hierarchical PhaseTree model in both C++ and GLSL. The source code for the models show in the paper is available at: https://github.com/Arches-Team/PhaseTree. Our system supports exporting construction trees as a set of GLSL functions, enabling real-time Sphere Tracing on graphics hardware. All examples presented in this paper were created on a desktop workstation equipped with Intel® Core *i*9, clocked at 5 GHz with 64 GB of RAM, and an NVidia GTX 4080 graphics card. The images were produced using Sphere Tracing in a Monte Carlo rendering shader.

7.1 Expressiveness and modeling power

The PhaseTree can represent a variety of objects, ranging from organic structures (Figure 1) to complex manufactured objects (Figure 17) or layered geological formations (Figure 27). One of the main advantages of the PhaseTree is its ability to represent complex multiphase objects in a unified manner within a single compact, memory efficient, construction tree. The combination (Section 4.1) and hybrid operators (Section 5) enable the definition of complex operations that act only on selected parts of an object, as illustrated throughout the paper, and particularly in Figures 4 and 18.

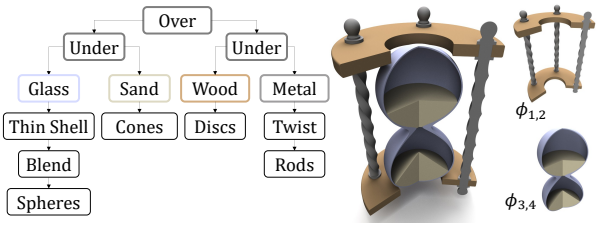

Fig. 18. An hourglass, modeled by assembling 4 different phases.

Figure 18 shows an hourglass modeled with 4 different phases. The creation of the glass bulb with sand inside deserves special attention and demonstrate the modeling power of the PhaseTree. A glass bulb volume was created as a signed distance field using a smooth union of spheres. This model was then converted into two phases using the two-phase thin shell operator (Section 4.1) to generate a glass envelope and define the interior as a temporary phase. This temporary phase in then intersected with a cone and box for modeling the sand inside the hourglass, exactly into contact with the glass. Maintaining precise contact between sand and glass would have been extremely complex without the multiphase operators. The wood discs and the metal rods are finally simply assembled with *under* and *over* operators.

Since all phases are defined consistently in the form of a signed distance field, they can be reused across different models or even leveraged for fabrication.

7.2 Performance

Compared to mono-phase models such as classical signed distance field construction trees, multiphase models are inherently more computationally demanding, as evaluating the field at a point $\mathbf{p}$ may require computing multiple phase functions. Specifically, instead of a single signed distance evaluation $f(\mathbf{p})$, a multiphase model requires evaluating the vector-valued field $\Phi(\mathbf{p}) = (\phi_1(\mathbf{p}), \ldots, \phi_n(\mathbf{p}))$.

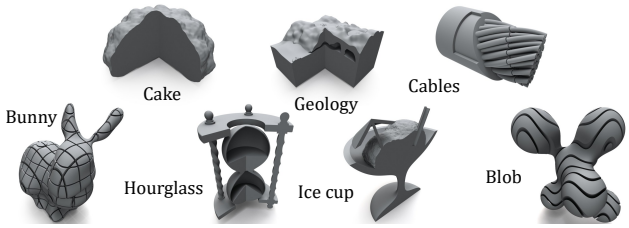

Fig. 19. Monolithic signed distance field objects used for computing the timings of Table 1.

Object	Figure	Monolithic		Multiphase	
		$\#\mathcal{N}$	Time (ms)	$\#\mathcal{N}$	Time (ms)
Bunny	26	6	5458	6	5414 (0%)
Cables	17	34	7361	27	9121 (+24%)
Hourglass	18	40	488	43	604 (+24%)
Geology	27	51	555	58	585 (+5%)
Cake	13	73	1158	81	1054 (−9%)
Blob	26	21	1687	21	1880 (+11%)
Ice cup	4	52	1041	47	816 (−22%)

Table 1. Comparison between multiphase traditional signed distance field models with identical geometries. Timings, reported in milliseconds (ms), were obtained by rendering the models both with and without a ground plane and secondary ray bounces. In some cases, multiphase models require fewer nodes ($\#\mathcal{N}$) than their signed distance field counterparts, as explicit phase information enables more concise trees. For the Bunny model, we computed the theoretical Lipschitz bound from the layer weights of the SIREN network, to guarantee a consistent 1-Lipschitz primitive.

We evaluated performance by rendering the models using Sphere Tracing with a simple Monte Carlo strategy, averaging the computation time over one ray per pixel. Table 1 reports comparative runtimes for objects with identical geometry, represented either as multiphase models or as standard signed distance field-based constructions (see Figure 19). The overhead introduced by the multiphase representation typically ranges between 5% and 25%.

In some rare cases, the multiphase models yield a more compact tree with fewer nodes, resulting in performance comparable to that of their mono-phase signed distance field counterparts. Indeed, the construction trees may differ slightly between mono-phase and multiphase representations, as the multiphase model introduces dedicated primitives and operators designed to explicitly encode phase information within different parts of an object.

Moreover, in all our examples, we present phase-specific cut views of the objects (e.g., the cable shown in Figure 17), a task that is significantly more challenging in a mono-phase setting. Reproducing an identical cut geometry without multiphase operators typically relies on two alternatives: either embedding cutting primitives directly

within the model construction, or applying a Boolean difference with a complex cutting object at the root of the signed distance field tree as a post-processing step. Both approaches present limitations compared to the multiphase formulation, either requiring manual integration and adjustment of the cutting geometry during modeling, or necessitating additional nodes to represent a potentially complex cutting shape (which partly explains the differences in node counts reported in Table 1). In contrast, our approach enables cuts to be applied at a later stage using simple per-phase geometries, which facilitates interactive modification of the cutting configuration while reducing both the number of nodes and the overall modeling effort.

As expected, timings (Table 1) indicate that multiphase models are slightly more computationally expensive than their mono-phase counterparts. This overhead is balanced by the fact that standard signed distance field models cannot implicitly represent internal phase structure, and therefore require more complex constructions to approximate equivalent multi-material objects.

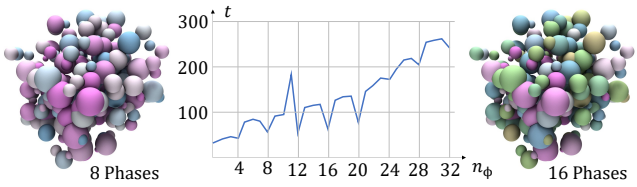

Fig. 20. Timings (in ms) show that the rendering time is almost proportional to the number of phases.

Figure 20 analyzes the computational overhead introduced by processing an increasing number of phases. As expected, the experiments show that rendering time grows approximately linearly with the number of phases, which is consistent with the number of operations performed by multiphase operators in the tree. In our current implementation, phase vectors Φ are stored and propagated using simple arrays. The performance curves reveal that the compiler significantly optimizes the generated code when the number of phases is a multiple of 4. Although we did not pursue a fully optimized GLSL implementation, additional performance gains could be achieved through straightforward optimizations, such as pruning inactive phases (i.e., $\phi_i = \infty$) during the evaluation of subtrees that involve only a limited subset of active phases.

In terms of memory consumption, the PhaseTree representation is comparable to standard signed distance field construction trees, and in several cases even more compact (*e.g.*, ice cup, cables), thanks to the expressive power of multiphase operators. Finally, the additional computations required to evaluate multiple phases are localized at internal nodes, as leaves and lower levels of the hierarchy typically correspond to standard signed distance field primitives.

7.3 Compatibility with signed distance fields algorithms

The PhaseTree model retains the same construction paradigm as existing signed distance field construction tree models. Instead of computing a single value $f(\mathbf{p}) \in \mathbb{R}$, it computes a vector value of n phases $\Phi(\mathbf{p}) \in \mathbb{R}^n$, each phase Φ describing a consistent signed distance field to Ω_i. Here we show how existing algorithms such as Sphere Tracing and field function queries acceleration methods can be seamlessly incorporated in the multiphase framework.

Sphere Tracing. The algorithm [Hart 1996] operates similarly as for standard signed distance fields with minimal adjustment. Consider a ray Δ parameterized by $\delta(t)$. We first identify the phase $\mu(\delta(0))$ at the ray origin, and then march adaptively along Δ until an interface with another phase is encountered. An interface is detected when the phase label changes, *i.e.* when $\mu(\delta(t)) \neq \mu(\delta(0))$, which is equivalently characterized by the ray exiting the initial phase: $\Phi_i(\delta(t)) > 0$.

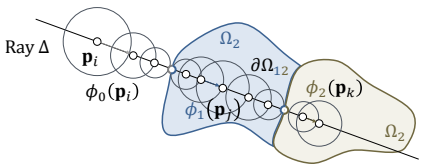

Fig. 21. Instead of using the signed distance fields to march along the ray, we use the smalest phase.

Marching is performed using a conservative step size given by the smallest absolute phase value: $\min_i |\Phi_i|$ (Figure 21). Since all phase functions Φ_i are guaranteed to be 1-Lipschitz, this choice provides a strict lower bound on the distance to any phase interface. As a result, the ray is guaranteed not to overstep interfaces, ensuring robust ray-interface intersection.

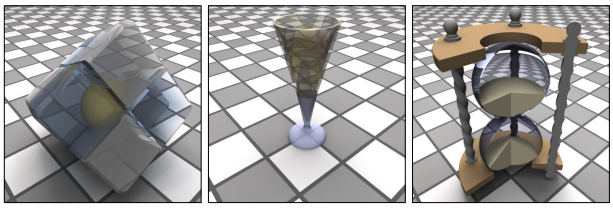

Fig. 22. Glass models with transparent phases, illustrating the consistent phase definition and unique negative phase function properties.

The algorithm terminates at the first encountered interface $\partial\Omega_{ij}$ between the initial phase Φ_i and a new phase Φ_j. At this location, the outgoing phase can be identified unambiguously, and the corresponding interface normals satisfy $\mathbf{n}_i = -\mathbf{n}_j$ at $\partial\Omega_{ij}$ (see Figure 3). These properties hold independently of the local configuration of phases, including contact lines or junction points between more than two phases and tightly nested interfaces (see Section 7.4). For physically based rendering, the coherent phase information encoded in the PhaseTree enables the application of appropriate reflection or refraction laws at each interface (Figure 22). Crucially, the Lipschitz property ensure that recursive ray spawning remains stable and free of self-intersections or missed interfaces.

Bounding volumes and proxies. Because of its hierarchical structure, the PhaseTree naturally lends itself for techniques accelerating field function queries. The bounding volumes and proxies techniques introduced in by Hubert-Brierre *et al.* [2025] can be inserted at any level in the tree to reduce the cost of the query $\Phi(\mathbf{p})$.

Automatic bounding volume computation is also possible with Sphere Carving [Schott et al. 2025] for example, either computing the bounding volume for any phase ϕ_i, of for the entire object using the merge operator (see Section 4.2). Figure 23 shows the result of Sphere Carving on the hourglass (merge of all phases), and on two of its phases, the twisted rods and the glass bulb. Once the phases have been extracted into a single standard signed distance field, the algorithm can be applied without any modification.

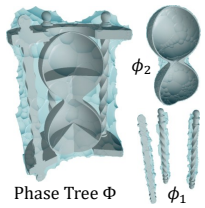

Fig. 23. Sphere carving.

7.4 Interface between phases

Because exactly one phase is negative at any point in space, the formulation naturally supports multiphase interfaces, including junctions between more than two phases.

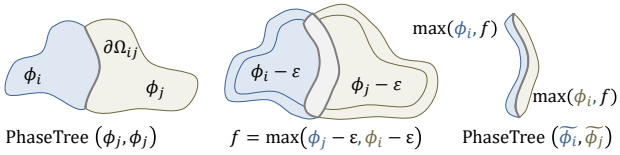

Fig. 24. Interface computation between two phases.

Consider a multiphase object $\mathcal{A}$, the signed distance field f representing the thin volume $\mathcal{S}$ embedding the interface $\partial\Omega_{ij}$ between two phases $\mathcal{A}_i$ and $\mathcal{A}_j$ can be defined as:

$$\mathcal{S} = I(\mathcal{A}_i, \varepsilon) \cap I(\mathcal{A}_j, \varepsilon)$$

The operator I denotes the inflation (see Section 4.1). The associated signed distance field f_{ij} can be computed as:

$$f = \max(\phi_{\mathcal{A}_i} - \varepsilon, \phi_{\mathcal{A}_j} - \varepsilon)$$

The function f is a signed distance field that can be used in the hybrid intersection operator (Section 5.1) to keep the existing materials and create a thin volume representing the multi-phase interface (see Figure 24). The two-phase representation of the thin interface can be defined as:

$$\Phi = \begin{cases} \phi_k = \max(f, \phi_{\mathcal{A}_k}) & \forall k \in \{i, j\} \\ \phi_k = \infty & \text{otherwise} \end{cases}$$

The complete set of interfaces between all phases is directly defined by computing the intersection between all inflated phases:

$$\Phi = \left(\max(f, \phi_{\mathcal{A}_k}), \forall k \in [1, n]\right)$$

$$f = \min_{(i,j)\in[1,n]^2, i\neq j} \max(\phi_{\mathcal{A}_i} - \varepsilon, \phi_{\mathcal{A}_j} - \varepsilon)$$

Figure 25 shows an example of a multiphase model composed of four distinct phases, along with the corresponding set of thin volumes representing the interfaces $\partial\Omega_{ij}$ between phases. The figure highlights thin surfaces separating pairs of phases, as well as contact lines and junctions where more than two phases meet.

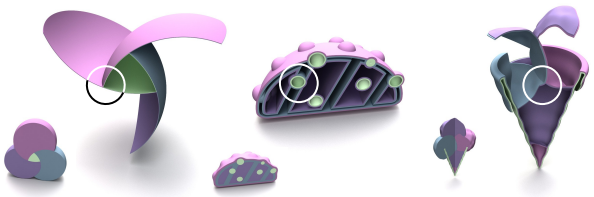

Fig. 25. Thin non-manifold interfaces extracted by the method, featuring triple and quadruple junction points.

7.5 Integration of other models

As the PhaseTree extends traditional signed distance field, it is also compatible with models that can be converted to conservative or 1-Lipschitz signed distance field models. Dedicated neural models for generating 1-Lipschitz signed distance field have been proposed [Coiffier and Béthune 2024], and are well suited for integration within our framework; however, such integration is left for future work and falls beyond the scope of this paper. Figure 26 shows examples featuring the Stanford Bunny obtained from a SIREN (neural representation using periodic activation functions) [Sitzmann et al. 2020] and a BlobTree [Wyvill et al. 1999] model representing seven spheres blended together. Details for converting a BlobTree to a 1-Lipschitz signed distance field are provided in Appendix 9.2.

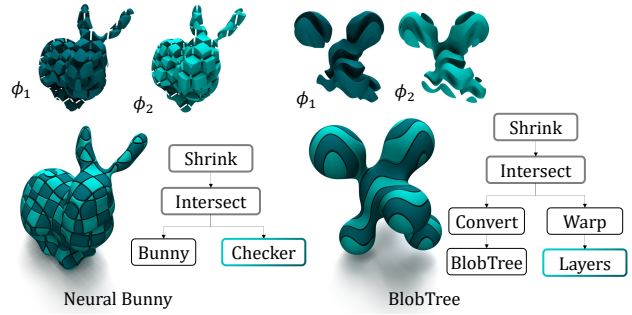

Fig. 26. A neural and BlobTree models decomposed into two separate phases using a checkered phase primitive.

Recall that a SIREN is a neural network architecture employing sinusoidal activation functions. Formally, the layers are defined using parameterized sine functions and the network is obtained by composing such layers. Consequently, the resulting signed distance field can be interpreted as a composition of Lipschitz functions, and is therefore λ-Lipschitz (see Appendix 9.4 for the derivation of the theoretical bound). In our experiments, we computed the theoretical bound λ and defined f/λ as the 1-Lipschitz function. Note that a tighter, practical bound of the maximum gradient magnitude λ can be obtained by sampling, rather than relying on the theoretical bound, as sinusoidal activations may locally compensate each other, albeit without formal guarantees. We also restrict the queries to the learned area by using a proxy sphere centered around the model.

In both cases, the signed distance field-based implicit models were used as volumes defining the overall geometry of the object (Figure 26). Separate phases were produced by intersecting these volumes with an infinite checker and set of layers multiphase primitive

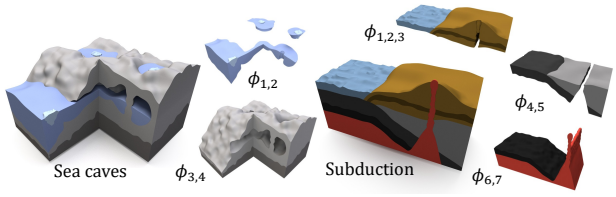

Fig. 27. Three dimensional terrains, featuring different volumetric information such as geological strata, faults, caves and karstic networks.

respectively (see Appendix 9.1 for implementation details), using the hybrid intersection operator (see Section 5). Naturally, such models can be used interchangeably with standard signed distance field nodes within any other PhaseTree.

7.6 Comparison to other models

A first, straightforward yet essential point of comparison is with solid-textured implicit surfaces [Wyvill et al. 1999]. Procedural solid textures can be viewed as functions $\tau : \mathbb{R}^3 \rightarrow \mathbb{R}^n$ that assign material attributes $\tau(\mathbf{p})$ to every point in space. In texture-tree approaches [Schmitt et al. 2001], the texture function τ is defined in a construction tree that is decoupled from the geometric representation. Consequently, the geometry remains monolithic, as distinct material phases cannot be identified, isolated, reused and manipulated explicitly during modeling.

In contrast, our approach explicitly encodes the geometry of each phase within the model as a well-defined 1-Lipschitz signed distance field. This tight coupling between geometry and phase information fundamentally changes the authoring workflow: individual phases can be created, edited, combined, or reused as first-class modeling primitives, rather than being defined through texture space.

Importantly, the PhaseTree remains fully compatible with solid texturing. Beyond compatibility, the availability of phase distance functions ϕ_i provides artists and procedural designers with additional geometric handles. These distance bounds can be exploited to drive texture blending, parameter modulation, or shading decisions in a spatially coherent manner using a different texture τ_i for each phase i. The PhaseTree allows for proximity-based transitions or phase-aware material variation, enabling workflows that are difficult or impossible to achieve with monophase implicit surface and texture-only representations.

Multiscale vector volumes [Wang et al. 2011] encode multiple phases using grid-based signed distance fields, organized either in a hierarchical binary tree or as piecewise polynomial functions defined over an adaptively subdivided octree. Although these representations are expressive and can approximate arbitrary input geometry, they rely on sampled fields and interpolation, which weakens their signed distance guarantees and limits their resolution independence. In practice, representing a multiphase object typically requires several megabytes of memory. By contrast, our procedural model represents complex multiphase geometry using compact hierarchical construction trees that require only a few kilobytes of storage, are fully resolution independent, and can be directly compiled into GLSL for efficient execution on graphics hardware.

Object-space multiphase implicit functions [Yuan et al. 2012] define piecewise polynomial fields over an adaptive octree, yielding a compact mathematical form for multiphase segmentation. However, the resulting fields are not guaranteed to be signed distance functions, nor to satisfy a global 1-Lipschitz condition. As a consequence, standard signed distance field-based algorithms such as Sphere Tracing, offsetting, or robust Boolean operations cannot be applied without additional corrections or reinitialization steps.

In contrast, our work introduces *multiphase signed distance fields*, in which phase information is integrated directly into a construction tree while preserving the signed distance property of each phase. All the operators are designed to maintain 1-Lipschitz continuity, ensuring full compatibility with classical signed distance field-based algorithms, including Sphere Tracing, spatial queries, and geometric transformations. Moreover, our model enables fine-grained selection, reuse, and recombination of arbitrary substructures through hybrid operators, providing a level of modeling flexibility that is difficult to achieve with grid-based multiphase representations.

Recently, Chen *et al.* [2025] introduced a method for reconstructing non-manifold surfaces by extracting consistent material interfaces from unsigned distance fields, resolving ambiguities near complex junctions. The approach robustly recovers sharp, topologically valid interfaces without requiring sign information. In contrast, multiphase signed distance field partitions space into multiple labeled regions with explicitly encoded phase information, thereby indirectly ensuring consistent, potentially non-manifold, interfaces between multiple phases. Unlike unsigned distance field reconstruction, our multiphase model relies on a procedural definition of volumes using signed distance fields with consistent interfaces.

7.7 Limitations

While our approach extends signed distance field with native multiphase capabilities, it also introduces several limitations. From a performance standpoint, increasing the number of phases directly impacts the cost of the multiphase field evaluations $\Phi(\mathbf{p})$ and, consequently, the efficiency of Sphere Tracing, as illustrated in Figure 20. This overhead can be partially mitigated by introducing efficient proxy nodes within the construction tree, at the cost of deeper hierarchies and additional traversal overhead.

Another limitation arises near interfaces $\partial\Omega_{ij}$ between phases $\phi_{\mathcal{A}_i}$ and $\phi_{\mathcal{A}_j}$, where the signed distance may become locally smaller. As a result, Sphere Tracing takes shorter steps in these regions, increasing the number of iterations required for convergence. In practice, we therefore discourage aggressively merging multiple phases into a single mono-phase signed distance field when the resulting geometry contains a large number of internal phase interfaces (Section 4.2), as this can significantly degrade ray-object intersection.

Finally, our current authoring workflow relies on scripting construction trees or manually composing signed distance field primitives using multiphase and hybrid operators. For example, the wires embedded in the cable shown in Figure 17 are procedurally placed within the insulating phase using a Fibonacci spiral on a disk, while surface bumps on the tentacle model are generated using a Fibonacci spiral on a sphere. While this is sufficient for procedural modeling

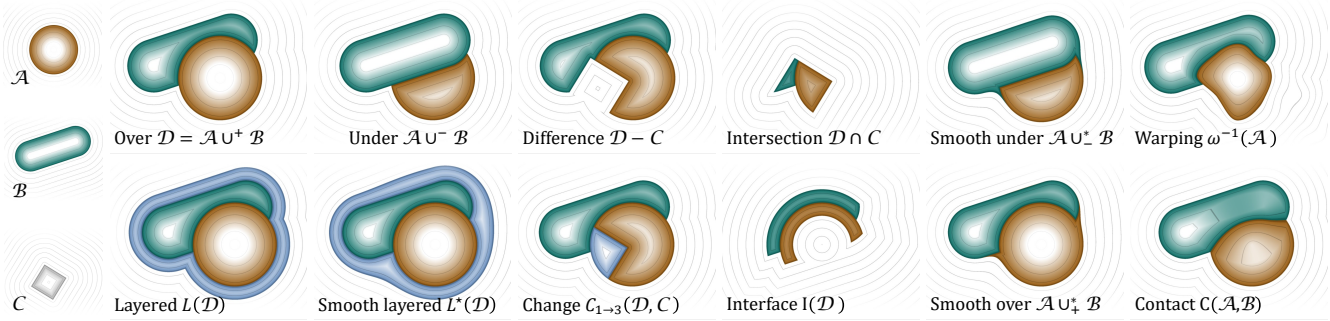

Fig. 28. Visual representation of some operators described throughout the paper: $\mathcal{A}$ and $\mathcal{B}$ represent two objects with their corresponding phase functions $\Phi_{\mathcal{A}}$ and $\Phi_{\mathcal{A}}$, whereas C represent a signed distance field, $\mathcal{D}$ is defined as $\mathcal{D} = \mathcal{A} \cup_{+} \mathcal{A}$.

or editing objects, authoring large complex scenes feature many objects with several hundred nodes remains challenging. These examples highlight the need for higher-level, interactive authoring tools. Extending recent interactive implicit modeling systems, such as the work of Riso *et al.* [2024], to support multiphase objects may be a promising direction for future research.

8 CONCLUSION AND FUTURE WORK

We introduced the PhaseTree model, a novel representation generalizing signed distance fields that enables the encoding of multiphase objects within a unified implicit framework. The proposed model is compatible with other implicit representations and inherits the compactness and resolution independence of procedurally defined implicit surfaces while extending them with native support for multiple materials or phases. Crucially, the construction operators are designed to guarantee the mathematical consistency and coherence of both the individual phases and the interfaces separating them.

Through a comprehensive set of operators for conversion, phase processing, and warping, the PhaseTree model supports a wide spectrum of modeling scenarios, ranging from manufactured artifacts to organic shapes and complex geological structures.

Future work will explore the generalization of operators and multiphase primitives, allowing for more complex combination between phases and enabling richer and more expressive representations. Inspired by the interactive modeling framework of Riso *et al.* [2024], we also plan to investigate interactive control mechanisms. In particular, we expect that explicitly representing phases will provide novel and powerful handles for user-driven modeling. Finally, the distance information associated with the different phases could be exploited to guide solid texturing and shading. For instance, rock or sand regions located near water phases could be automatically interpreted as wet and shaded accordingly, enabling more physically plausible and context-aware appearance modeling.

ACKNOWLEDGMENTS

This work is funded by the projects EOLE ANR-23-CE56-0008 and MultIFOrm ANR-22-CE33-0018, supported by Agence Nationale de la Recherche (France).

REFERENCES

Wilhem Barbier, Mathieu Sanchez, Axel Paris, Elie Michel, Thibaud Lambert, Tamy Boubekeur, Mathias Paulin, and Theo Thonat. 2025. Lipschitz Pruning: Hierarchical Simplification of Primitive-Based SDFs. *Computer Graphics Forum* 44, 2 (2025), e70057.

James F. Blinn. 1982. A Generalization of Algebraic Surface Drawing. *ACM Transactions on Graphics* 1, 3 (1982), 235–256.

Jules Bloomenthal and Ken Shoemake. 1991. Convolution surfaces. *SIGGRAPH Computer Graphics* 25, 4 (1991), 251–256.

Jules Bloomenthal and Brian Wyvill (Eds.). 1997. *Introduction to Implicit Surfaces.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Xuhui Chen, Fei Hou, Wencheng Wang, Hong Qin, and Ying He. 2025. MIND: Material Interface Generation from UDFs for Non-Manifold Surface Reconstruction. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems.*

Guillaume Coiffier and Louis Béthune. 2024. 1-Lipschitz Neural Distance Fields. *Computer Graphics Forum* 43, 5 (2024), e15128.

Barbara Cutler, Julie Dorsey, Leonard McMillan, Matthias Müller, and Robert Jagnow. 2002. A procedural approach to authoring solid models. *ACM Transactions on Graphics* 21, 3 (2002), 302–311.

Daniel Dekkers, Kees Van Overveld, and Rob Golsteijn. 2004. Combining CSG Modeling with Soft Blending Using Lipschitz-Based Implicit Surfaces. *The Visual Computer* 20, 6 (2004), 380–391.

David S. Ebert, F. Kenton Musgrave, Darwyn Peachey, Ken Perlin, and Steve Worley. 2002. *Texturing & Modeling: A Procedural Approach.* San Francisco, CA, USA, Morgan Kaufmann.

Sarah F. Frisken, Ronald N. Perry, Alyn P. Rockwood, and Thouis R. Jones. 2000. Adaptively Sampled Distance Fields: A General Representation of Shape for Computer Graphics. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH'00).* 249–254.

Marie-Paule Gascuel. 1993. An implicit formulation for precise contact modeling between flexible solids. In *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '93).* Association for Computing Machinery, New York, NY, USA, 313–320.

John C. Hart. 1996. Sphere Tracing: A Geometric Method for the Antialiased Ray Tracing of Implicit Surfaces. *The Visual Computer* 12, 10 (1996), 527–545.

Samuel Hornus, Alexis Angelidis, and Marie-Paule Cani. 2003. Implicit modelling using subdivision curves. *The Visual Computer* 19, 2-3 (2003), 94–104.

Pierre Hubert-Brierre, Adrien Peytavie Eric Guérin, and Eric Galin. 2025. Accelerating Signed Distance Functions. *Computer Graphics Forum* 45, 7 (2025), e70258.

Frank Losasso, Tamar Shinar, Andrew Selle, and Ronald Fedkiw. 2006. Multiple interacting liquids. *ACM Transactions on Graphics* 25, 3 (2006), 812–819.

Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. 2022. Instant Neural Graphics Primitives with a Multiresolution Hash Encoding. *ACM Transactions on Graphics* 41, 4, Article 102 (2022), 15 pages.

Ken Museth, David Breen, Ross Whitaker, and Alan Barr. 2002. Level Set Surface Editing Operators. *ACM Transactions on Graphics* 21, 3 (2002), 330–338.

Ken Museth, David E. Breen, Ross T. Whitaker, Sean Mauch, and David Johnson. 2005. Algorithms for Interactive Editing of Level Set Models. *Computer Graphics Forum* 24, 4 (2005), 821–841.

Alexandrina Orzan, Adrien Bousseau, Holger Winnemöller, Pascal Barla, Joëlle Thollot, and David Salesin. 2008. Diffusion curves: a vector representation for smooth-shaded images. *ACM Transactions on Graphics* 27, 3 (2008), 1–8.

Alexander Pasko, Valery Adzhiev, Alexei Sourin, and Vladimir Savchenko. 1995. Function representation in geometric modeling: concepts, implementation and applications. *The Visual Computer* 11, 8 (1995), 429–446.

Marzia Riso, Élie Michel, Axel Paris, Valentin Deschaintre, Mathieu Gaillard, and Fabio Pellacini. 2024. Direct Manipulation of Procedural Implicit Surfaces. *ACM Transaction on Graphics* 43, 6, Article 238 (2024), 12 pages.

B. Schmitt, A. Pasko, V. Adzhiev, and C. Schlick. 2001. Constructive texturing based on hypervolume modeling. *The Journal of Visualization and Computer Animation* 12, 5 (2001), 297–310.

Hugo Schott, Theo Thonat, Thibaud Lambert, Eric Guérin, Eric Galin, and Axel Paris. 2025. Sphere Carving: Bounding Volumes for Signed Distance Fields. *ACM Transaction on Graphics* 44, 4, Article 137 (2025), 13 pages.

Andrei Sherstyuk. 1999. Kernel functions in convolution surfaces: A comparative analysis. *The Visual Computer* 15, 4 (1999), 171–182.

Vincent Sitzmann, Julien N.P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. 2020. Implicit Neural Representations with Periodic Activation Functions. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*. Curran Associates Inc., 7462–7473.

Kenshi Takayama, Olga Sorkine, Andrew Nealen, and Takeo Igarashi. 2010. Volumetric modeling with diffusion surfaces. In *ACM SIGGRAPH Asia*. Association for Computing Machinery, New York, NY, USA, Article 180, 8 pages.

Thibault Tricard, Semyon Efremov, Cédric Zanni, Fabrice Neyret, Jonàs Martínez, and Sylvain Lefebvre. 2019. Procedural phasor noise. *ACM Transactions on Graphics* 38, 4, Article 57 (2019), 13 pages.

Lvdi Wang, Yizhou Yu, Kun Zhou, and Baining Guo. 2011. Multiscale vector volumes. *ACM Transactions on Graphics* 30, 6 (2011), 1–8.

Lvdi Wang, Kun Zhou, Yizhou Yu, and Baining Guo. 2010. Vector solid textures. *ACM Transactions on Graphics* 29, 4, Article 86 (2010), 8 pages.

Steven Worley. 1996. A cellular texture basis function. In *Proceedings of the Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '96)*. Association for Computing Machinery, New York, NY, USA, 291–294.

Brian Wyvill, Andrew Guy, and Eric Galin. 1999. Extending the CSG Tree - Warping, Blending, and Boolean Operations in an Implicit Surface Modeling System. *Computer Graphics Forum* 18, 2 (1999), 149–158.

Geoff Wyvill, Craig McPheeters, and Brian Wyvill. 1986. Data Structure for Soft Objects. *The Visual Computer* 2 (1986), 227–234.

Zhan Yuan, Yizhou Yu, and Wenping Wang. 2012. Object-space multiphase implicit functions. *ACM Transactions on Graphics* 31, 4, Article 114 (2012), 10 pages.

Cédric Zanni, Adrien Bernhardt, Maxime Quiblier, and Marie-Paule Cani. 2013. SCALe-invariant Integral Surfaces. *Computer Graphics Forum* 32, 8 (2013), 219–232.

Hong-Kai Zhao, T. Chan, B. Merriman, and S. Osher. 1996. A Variational Level Set Approach to Multiphase Motion. *Journal of Computational Physics* 127, 1 (1996), 179–195.

9 APPENDIX

9.1 Infinite multiphase primitives

The infinite checker multiphase object is defined as the union of cubes of two different phases. Let $\lfloor \mathbf{p} \rfloor$ denote the vector with the floor function applied to its components. We define the three dimensional multiphase checker signed distance field by computing the fractional part $\mathbf{q} = \mathbf{p} - \lfloor \mathbf{p} \rfloor$ of the components of $\mathbf{p}$:

$$\Phi = \begin{cases} \phi_i = s \, \min(\mathbf{q}, 1 - \mathbf{q}) \\ \phi_j = -\phi_i \end{cases} \qquad s = \begin{cases} +1 & \text{if } \lfloor \mathbf{p} \rfloor \cdot (1,1,1) \text{ even} \\ -1 & \text{otherwise} \end{cases}$$

Similarly, the infinite multiphase layers primitive is defined as the set of parallel slices orthogonal to a given prescribed direction $\mathbf{u}$. Let $u = \mathbf{p}\dot{\mathbf{u}}$, we compute the fractional part $z = u - \lfloor u \rfloor$ and define the space filling two-phase primitive as:

$$\Phi = \begin{cases} \phi_i = s \, \min(\mathbf{q}, 1 - \mathbf{q}) \\ \phi_j = -\phi_i \end{cases} \qquad s = \begin{cases} +1 & \text{if } \lfloor u \rfloor \text{ even} \\ -1 & \text{otherwise} \end{cases}$$

9.2 BlobTree

Let $b : \mathbb{R}^3 \to \mathbb{R}$ denote the scalar field function of a BlobTree. Recall that $b(\mathbf{p}) = 0$ if $\mathbf{p}$ is outside of the compact support Ω of the object, and positive otherwise. The surface is defined as the set $\{b(\mathbf{p}) = T, \mathbf{p} \in \mathbb{R}^3\}$. Let λ denote the Lipschitz bound of b, the expression $(T - b(\mathbf{p}))/\lambda$ correctly defines a signed distance bound. However, $\forall \mathbf{p} \in \mathbb{R} - \Omega$, we have $b(\mathbf{p}) = 0$: the distance bound to the surface is constant and equal to T/λ outside of Ω, which yields small values outside and slows algorithms such as Sphere Tracing.

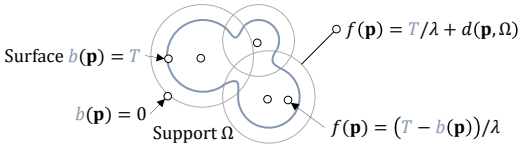

Fig. 29. Distance to a Blob model.

Therefore, we use the distance to the boundary of the domain $d(\mathbf{p}, \Omega)$ when $\mathbf{p} \notin \Omega$ which can be easily computed when traversing the BlobTree that combines compactly supported primitives built from skeletal elements. We define:

$$f(\mathbf{p}) = \begin{cases} (T - b(\mathbf{p}))/\lambda & \text{if } \mathbf{p} \in \Omega \\ T/\lambda + d(\mathbf{p}, \Omega) & \text{otherwise} \end{cases}$$

By construction, the function d is continuous, 1-Lipschitz, provides a lower bound to the Euclidean distance to the object, and allows for large steps when performing Sphere Tracing even when $\mathbf{p} \notin \Omega$.

9.3 Lipschitz property of smooth union

Let a and b denote two 1-Lipschitz signed distance fields. Let $d(a, b) = 1 - |b - a|/r$ if $|b - a| < r$ and 0 otherwise, we define the smooth union introduced in [Dekkers et al. 2004]:

$$s(a, b) = \min(a, b) - \frac{1}{6} r \left(\max \left(d(a, b), 0 \right) \right)^3$$

The problem is symmetric with respect to a and b (see Figure 30). We need to check the continuity of the gradient when $b - a \to r$ and $b - a \to 0$. Consider the left and right gradients at $b - a = r$, on the right $\nabla s = \nabla a$ and on the left $\nabla s = (1 - d^2/2)\nabla a + d^2/2 \nabla b$. Therefore both left and right limits of ∇s when $b - a \to r$ are equal to ∇a.

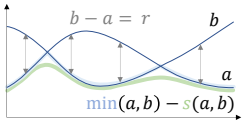

Fig. 30. Smooth union $s(a, b)$.

We also need to check continuity when $b - a \to 0$. On the left, if $a < b$ $\nabla s = (1 - d^2/2)\nabla b + d\nabla a$ and on the right for $a > b$, $\nabla s = (1 - d^2/2)\nabla a + d^2/2 \nabla b$. When $b - a \to 0$, $d \to 1$ and both derivatives converge to the same limit $(\nabla a + \nabla b)/2$. Therefore, s is C^1. Moreover, since $0 \leq d(a, b) \leq 1$, ∇s interpolates ∇a and ∇b and since a and b are 1-Lipschitz, then $\|\nabla s\| \leq 1$, and s is 1-Lipschitz.

9.4 Lipschitz bound of SIREN model

Recall that the layers are defined using parameterized sine functions $\sin(w_i \, \mathbf{p} + b_i)$, where w_i and b_i denote the weights and biases, respectively, and the signed distance field is defined as the composition $f = \bigcirc_{j=i}^{n} f_j$. The gradient and a theoretical Lipschitz bound are:

$$\nabla f = \nabla \bigcirc_{i=1}^{n} f_i = \prod_{i=1}^{n} \nabla f_i \bigcirc_{j=i}^{n} f_j \qquad \lambda \leq \prod_{i=1}^{n} w_i$$

In our experiments (Table 1), we used the theoretical bound λ computed from the weights of the layers.