Computers & Graphics 35 (2011) 596–603

Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

SMI 2011: Full Paper

Interactive modeling of implicit surfaces using a direct visualization approach with signed distance functions

Tim Reiner [a,*], Gregor Mückl [b], Carsten Dachsbacher [a]

[a] Computer Graphics Group, Karlsruhe Institute of Technology, Germany
[b] SimTech, University of Stuttgart, Germany

ARTICLE INFO

Article history:
Received 11 December 2010
Received in revised form
13 February 2011
Accepted 9 March 2011
Available online 21 March 2011

Keywords:
Distance functions
Implicit surfaces
Implicit surface rendering
Interactive modeling

ABSTRACT

Modeling appealing virtual scenes is an elaborate and time-consuming task, requiring not only training and experience, but also powerful modeling tools providing the desired functionality to the user. In this paper, we describe a modeling approach using signed distance functions as an underlying representation for objects, handling both conventional and complex surface manipulations. Scenes defined by signed distance functions can be stored compactly and rendered directly in real-time using sphere tracing. Hence, we are capable of providing an interactive application with immediate visual feedback for the artist, which is a crucial factor for the modeling process. Moreover, dealing with underlying mathematical operations is not necessary on the user level. We show that fundamental aspects of traditional modeling can be directly transferred to this novel kind of environment, resulting in an intuitive application behavior, and describe modeling operations which naturally benefit from implicit representations. We show modeling examples where signed distance functions are superior to explicit representations, but discuss the limitations of this approach as well.

1. Introduction

Implicit descriptions of smooth continuous surfaces are well established in computer graphics, next to discrete approximations like the ubiquitous polygonal meshes. Implicit surfaces spark interest due to their intrinsic way of representing solid objects, aptitude for constructive solid geometry, and eminent blending abilities.

However, it is inherently difficult to render implicit surfaces directly in real-time. This requires an extensive incremental search for surface points, which are neither explicitly given nor easy to compute. Consequently, indirect visualization techniques, such as marching cubes [7], became popular, which generate discrete approximations of surfaces first, making them fast to render. However, these intermediate steps introduce geometric aliasing, giving up both smoothness and high-frequency details.

Implicit surfaces are amenable to ray tracing, but it is difficult to find intersections of rays with complex surfaces analytically. Thus, ray marching is often used to render implicit surfaces directly, where possible intersections are determined by marching along a ray in small steps until a surface is hit. Sphere tracing [4] speeds up ray marching significantly, but requires distance surfaces [1], implicitly described by functions that return a measurement or bound of the geometric distance, instead of a generic algebraic distance.

It has always been tempting and desirable to use implicit surfaces for shape modeling. Providing an interactive modeling environment for this purpose is difficult, as it involves the necessity to render implicit surfaces in real-time in order to provide constant visual feedback. Indirect visualization techniques are capable of doing so, but introduce another problem: while editing a surface, continuous re-evaluations of its discrete approximation are required. Unless this is achieved in real-time, interactivity gets interrupted at this crucial point. Reducing approximation quality helps, but it removes subtle details and introduces artifacts as well.

In this paper, we make the following contributions:

- We compose complex implicit surfaces using signed distance functions. Then, we show how to translate these compositions into a fragment shader that renders defined surfaces directly on graphics hardware. As a result of using signed distance functions, we are able to apply sphere tracing to efficiently render a scene in real-time.
- We present a complete modeling environment for objects and scenes that are implicitly defined by signed distance functions. As we are able to render them in real-time, we provide full interactivity for the modeling tool. We are neither dependent on indirect visualization techniques nor restricted by spatial limitations. Likewise, explicit grids or approximations are not required.

* Corresponding author.
E-mail address: tim.reiner@kit.edu (T. Reiner).

0097-8493/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cag.2011.03.010

2. Related work

Previously, a lot of pioneer work in implicit shape modeling has been done. In this section, we focus on fundamental and most related work, ranging from the notion and rendering of implicit surfaces to editing operators and modeling environments for them.

Groundbreaking, hypertextures [15] combined implicit shapes with textures to model various phenomena, extending the notion of shape by including surrounding volumes. Turk and O'Brien [23,24] introduced new shape transformations and interpolation techniques for implicit surfaces.

Loop and Blinn [6] showed how to render implicit surfaces in real-time on the GPU using analytic techniques for solving for roots of polynomials. Hence, they are limited to fourth order algebraic surfaces, and form objects by assembling piecewise smooth algebraic surfaces, which were introduced by Sederberg [20].

It becomes cumbersome to evaluate functions defining implicit surfaces of increasing complexity. Consequently, distance fields attracted attention, which explicitly store distance information inside a three-dimensional grid. Again, this leads to geometric aliasing in trade for fast distance evaluations. Obviously, this representation requires a tremendous amount of memory if used naively, and therefore substantially limits the possible extent of a scene. Frisken et al. [3] proposed adaptively sampled distance fields, which significantly reduce memory consumption for detailed objects.

The level set method, first presented by Osher and Sethian [12] and thoroughly explained by Osher and Fedkiw [13], defines a surface using a time-varying scalar function. It spawned a large body of research and gained recognition in modeling surface deformations.

Museth et al. [10] contributed editing operators for implicit surfaces based on level set models. Surfaces are imported as distance fields into their level set framework, allowing arbitrary surfaces modified with a single procedure. As their framework is based on uniform grids, it suffers from technical limitations again, particularly spatial limits, memory constraints, and artifacts due to discretization.

Comprehensive approaches to interactive implicit modeling were proposed. A primary contribution is HyperFun [16] that has been extended by others in various directions. In its essential form, it provides a symbolic user interface for direct textual input in a high-level geometric language. Note that interactivity does not extend to real-time surface visualization.

Our approach closely resembles BlobTrees [25], which have been widely adapted. Its hierarchical structure stores implicit surfaces at the leaves and places composition operators at internal nodes. ShapeShop [19] is a prominent example based on the BlobTree. Interactivity is achieved by introducing spatial caching for objects.

WarpCurves [22], inspired by FiberMesh [11] and Wires [21], extends ShapeShop to allow for interactive manipulation of implicit surfaces using explicit curve-based spatial deformations. Recently, Martinez Esturo et al. [9] described a method for continuous deformations of implicit surfaces focusing on volume, continuity, and topology conservation.

Related important areas of research in interactive modeling are haptics-based [5] and sketch-based [19] user interfaces for convenient sculpting.

Note that the modeling environments mentioned before use an indirect marching cubes visualization, contrary to our real-time direct visualization of the scene.

3. Describing scenes with SDFs

This section briefly introduces SDFs (signed distance functions), shows how to describe primitives with them, and how transformations and combinations can be applied in order to create complex objects. Eventually, whole scenes emerge by assembling several objects.

3.1. Definition

We mostly stick to the notation from [13], which provides, inter alia, a comprehensive introduction to implicit surfaces. Let Ω^- and Ω^+ be the space in and outside of an implicit surface, respectively, and $\partial\Omega$ the interface in between, i.e., the set of points composing the surface. A signed distance function $\phi(\vec{x})$ is defined as

$$\phi(\vec{x}) = \begin{cases} \min(|\vec{x}-\vec{x}_I|) & \text{if } \vec{x} \in \Omega^+, \\ 0 & \text{if } \vec{x} \in \partial\Omega, \\ -\min(|\vec{x}-\vec{x}_I|) & \text{if } \vec{x} \in \Omega^- \text{ for all } \vec{x}_I \in \partial\Omega, \end{cases}$$

and returns the Euclidean distance from $\vec{x}$ to its closest surface point, with a negative sign if inside. Thus, SDFs are a subset of implicit functions, which rather imply algebraic distances.

3.2. Properties

SDFs retain characteristics of implicit functions, yet provide more beneficial properties, such as $|\nabla\phi| = 1$. This is comprehensible, since ϕ is Euclidean distance, and satisfies the criteria of sphere tracing precisely; see Section 4.2. Note that the equation is not true for points equidistant to at least two interface points, where the path of steepest descent is ambiguous. Fortunately, we can safely ignore this issue, as we march along given ray directions only.

We are interested in providing a comprehensive variety of flexible modeling operations and transformations. In exchange, we comply with producing distorted distance functions that do not yield the accurate Euclidean distance anymore. If $0 < |\nabla\phi| < 1$, sphere tracing decelerates but still ensures not to penetrate surfaces. Alas, this is not the case for $|\nabla\phi| > 1$, but sphere tracing is robust enough if a bound is known. Section 4.2 presents a concept to compensate for gradients too steep.

As a matter of particular interest, SDFs remain monotonic across the interface, not having a kink like (unsigned) distance functions. This allows reliable gradient constructions, e.g., using central differences, which is required for illumination and shading.

3.3. Describing primitives

Many primitives can be described directly using an SDF. For example, $\phi(\vec{x},r) = |\vec{x}| - r$ describes a sphere at the origin with radius r. Other basic primitives such as tori and infinite planes, cylinders, and cones are easy to describe as well.

Given the SDF of a two-dimensional curve, it can be easily extended to a solid of revolution. Distances are evaluated by projecting points in question onto the plane where the curve is defined.

Primitives featuring hard edges between surface regions require SDFs to have corresponding kinks. This can be accomplished by composing individual SDFs for each surface region, using case-by-case analysis whose region is closest to a point in question. We try to avoid case-by-case analysis in our shaders, however, as branching is still an expensive operation on the GPU. Instead, we edge surfaces by switching function spaces. For instance, a sphere formed using the L_∞ metric represents an axis-aligned cube. In this way, we can state an SDF for a box with width w, height h, and depth d as

$$\phi(\vec{x},w,h,d) = \max(|\vec{x}_1| - w/2, |\vec{x}_2| - h/2, |\vec{x}_3| - d/2).$$