Available online at www.sciencedirect.com
ScienceDirect
Computers & Graphics 31 (2007) 698–709
COMPUTERS & GRAPHICS
www.elsevier.com/locate/cag

Technical Section

Progressive refinement rendering of implicit surfaces

Manuel N. Gamito*,[1], Steve C. Maddock

Department of Computer Science, The University of Sheffield, UK

Received 29 March 2006; received in revised form 12 March 2007; accepted 20 April 2007

Abstract

The visualisation of implicit surfaces can be an inefficient task when such surfaces are complex and highly detailed. Visualising a surface by first converting it to a polygon mesh may lead to an excessive polygon count. Visualising a surface by direct ray casting is often a slow procedure. In this paper we present a progressive refinement renderer for implicit surfaces that are Lipschitz continuous. The renderer first displays a low resolution estimate of what the final image is going to be and, as the computation progresses, increases the quality of this estimate at an interactive frame rate. This renderer provides a quick previewing facility that significantly reduces the design cycle of a new and complex implicit surface. The renderer is also capable of completing an image faster than a conventional implicit surface rendering algorithm based on ray casting.

Keywords: Progressive refinement; Ray casting; Implicit surfaces; Lipschitz bounds

1. Introduction

Implicit surfaces find application in many areas of Computer Graphics where objects exhibiting complex topologies, i.e. with many holes or disconnected pieces, need to be modelled. An implicit surface is defined as the set of all points $\mathbf{x} \in \mathbb{R}^3$ that verify the condition $f(\mathbf{x}) = 0$ for some function $f: \mathbb{R}^3 \to \mathbb{R}$. Modelling with implicit surfaces amounts to the construction of an appropriate function f, called the *implicit function*, that will generate the desired surface. Over the years, three main strategies for the design of implicit functions have become established. *Algebraic surfaces* arise from the use of polynomial implicit functions [1]. Sums of radial basis functions are also a popular method of constructing implicit surfaces. Depending on the choice of radial basis function, these can be called *blobby models* [2], *metaballs* [3] or *soft objects* [4]. By carefully selecting the weights associated with each radial basis function, it is also possible to have an *interpolating implicit surface* that is constrained to pass through a set of scattered data points [5,6]. Finally, *hypertextures* are an example of implicit functions that are generated by perturbing the surface of an initially smooth object with a combination of procedural noise functions [7]. By summing together many such noise functions a fractal hypertexture can be generated, having an associated fractal dimension.

Rendering algorithms for implicit surfaces can be broadly divided into meshing algorithms and ray casting algorithms. Meshing algorithms convert an implicit surface to a polygonal mesh format, which can be subsequently rendered in real time with modern graphics processor boards [8–10]. Ray casting algorithms bypass mesh generation entirely and compute instead the projection of an implicit surface on the screen by casting rays from each pixel into three-dimensional space and finding their intersection with the surface [11–13]. We propose an extension to ray casting algorithms for implicit surfaces that incorporates a progressive refinement rendering principle. The idea of progressive refinement for image rendering was first formalised in 1986 [14]. Progressive refinement rendering has received much attention in the fields of radiosity and global illumination [15–17]. Progressive refinement approaches to volume rendering have also been developed [18,19]. Our implicit surface renderer

*Corresponding author.
E-mail address: mag@dcs.shef.ac.uk (M.N. Gamito).
[1]Supported by Grant SFRH/BD/16249/2004 from Fundação para a Ciência e a Tecnologia, Portugal.

0097-8493/$ - see front matter © 2007 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cag.2007.04.011

uses progressive refinement to visualise an increasingly better approximation to the final implicit surface. It allows the user to make quick editing decisions without having to wait for a full ray casting solution to be computed. Because the algorithm is progressive, the rendering can be terminated as soon as the user is satisfied or not with the look of the surface. The renderer can also be interactively controlled by the user through the specification of image regions. Image refinement will only occur inside a region, once the region becomes active, while the remainder of the image is kept on hold. In this way the user can steer the application into rendering image regions that he considers to be more interesting or troublesome.

Our method is able to render any implicit surface whose generating function f is *Lipschitz continuous*. For a function to be Lipschitz continuous there must exist a real number λ such that

$$|f(\mathbf{x}_a) - f(\mathbf{x}_b)| < \lambda \|\mathbf{x}_a - \mathbf{x}_b\| \quad \text{for any } \mathbf{x}_a, \mathbf{x}_b \in \mathbb{R}^3. \tag{1}$$

A value $\lambda > 0$ that verifies (1) is called a *Lipschitz bound* of f. Any other value greater than λ also verifies (1) and is also a Lipschitz bound. The smallest of all these Lipschitz bounds is called the *Lipschitz constant* of f. Convergence of the progressive refinement algorithm towards the final image depends on the value of λ that has to be provided beforehand and is specific to the particular implicit function being visualised. If the Lipschitz constant of f is known, the algorithm will have optimal convergence. Otherwise, a Lipschitz bound must be provided, with the algorithm exhibiting slower convergence, the larger the value of λ is. Values of λ smaller than the Lipschitz constant can also be attempted to increase the convergence rate at the price that surface visualisations are no longer guaranteed to be correct. Most implicit surfaces of interest in Computer Graphics are continuous, which implies that they are Lipschitz continuous also. The application of the progressive refinement renderer to Lipschitz continuous surfaces only is, therefore, not overly restrictive. Examples of surfaces that cannot be rendered with our proposed method can be found mainly within some types of algebraic surfaces, which contain isolated points where the surface gradient is infinite.

The main stage of our method consists in the subdivision of the image space into progressively smaller square samples. Information about the part of the implicit function that is visible through a sample is obtained by shooting a ray through the centre of the sample and marching towards the surface intersection point with the help of a guaranteed ray–surface intersection algorithm [13]. The surface Lipschitz bound is used to compute step lengths that bring the ray progressively closer to the surface. Projecting the area of a sample from the viewpoint and into object space defines the sample's view volume, which features the ray along its main axis. We enhance the surface intersection algorithm by testing for intersections inside the whole view volume of the sample whose ray is being traced. As we march along a ray towards the surface there comes a distance after which it is no longer possible to guarantee that no intersections occur inside the view volume that is defined around the ray. At this point, sample subdivision takes place and new rays are shot, each surrounded by a thinner view volume. The sample subdivision mechanism stops once a sample has reached pixel size, at which point conventional ray casting is used to march along the remainder of the distance towards the surface.

Image rendering takes place simultaneously with sample subdivision and ray casting so that, at any given time, the image shows the best approximation to the correct surface visualisation. Each sample's colour is obtained by evaluating a shading model at the point that corresponds to the current distance traced along the ray that passes through the centre of the sample. The shading accuracy improves as the distance along the ray converges towards the surface. This rendering model generates the visual effect of an implicit surface that is perceived to be shrinking towards its final configuration. The surface shrinking effect is a result of the set of image samples being subdivided and their rays being marched in parallel towards the surface. The previewing capability of the progressive refinement renderer is a consequence of the observation that early enlarged surfaces look already similar to the final implicit surface.

Section 2 describes previous work that provides previewing facilities for implicit surfaces. Some of the work described in that section was not developed with quick previewing in mind but it can be used to that effect. Section 3 describes our progressive refinement previewer. Although our main research focus is in the area of procedural landscape modelling with hypertextured surfaces, we show in Section 4 progressive refinement rendering examples for the three main categories of implicit surfaces: algebraic surfaces, surfaces generated from sums of radial basis functions and hypertextures. A performance comparison is also presented between our previewer and a standard rendering algorithm for implicit surfaces based on ray casting. Section 5 presents our conclusions. Appendix A describes an extension of the progressive refinement algorithm to incorporate anti-aliasing.

2. Previous work

One of the best known techniques for previewing implicit surfaces at interactive frame rates is based on the dynamic placement of discs that are tangent to the surface [20,21]. The discs are kept apart by the application of repulsive forces and are constrained to remain on the surface. Each disc is also made tangent to the surface by sharing the surface normal at the point where it is located. This previewing system relies on a characteristic of our visual system whereby we are able to infer the existence of an object based solely on the distribution of a small number of features on the surface of that object [22]. This visual trait only works, however, when the surface of the object is