Computers & Graphics 35 (2011) 148–159

Contents lists available at ScienceDirect

Computers & Graphics

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

Extended papers from NPAR 2010

Vector graphics depicting marbling flow

Ryoichi Ando [a,*], Reiji Tsuruno [b]

[a] Graduate School of Design, Kyushu University, 4-9-1, Shiobaru, Minamiku, Fukuokashi 815-8540, Japan
[b] Faculty of Design, Kyushu University, 4-9-1, Shiobaru, Minamiku, Fukuokashi 815-8540, Japan

ARTICLE INFO

Article history:
Received 29 August 2010
Received in revised form
31 October 2010
Accepted 5 November 2010
Available online 23 November 2010

Keywords:
Marbling
Surface flow
Front tracking

ABSTRACT

We present an efficient framework for generating marbled textures that can be exported into a vector graphics format based on an explicit surface tracking method. The proposed method enables artists to create complex and realistic marbling textures that can be used for design purposes. Our algorithm is unique in that the marbling paint on the surface of water is represented as an enclosed contour and is advected by fluid flow to deform the marbling silhouette. In contrast to previous methods, in which the shape is tracked with a concentration density field in Eulerian grids, our approach facilitates greater complexity that is free from grid resolution and per-pixel computation while retaining real-time performance. To forestall the propagation of large vertices, we adaptively resample the contours, exploiting the curvature and the turbulence of the fluid as criteria. At the convection phase, we parallelly advect contour particles on a Graphics Processing Unit (GPU) in addition to applying volume corrections. Finally, we quickly remove extremely thin lines in shapes to remove dozens of vertices. We performed our method with an interactive prototype to demonstrate the robustness of the proposed method in several scenarios.

1. Introduction

Marbling is a traditional technique that is used for decorating papers with paints floating on a liquid. Marbled patterns are created by dropping paints onto the surface of water and stirring the surface with brushes. The colors are then transferred to a sheet of paper by laying the paper on the surface of the water. Today, due to its ease of use and the vivid and unique patterns it produces, the marbling designs are printed worldwide onto various media, such as booklets and tissue boxes [18,13].

One of the most challenging aspects of simulating marbling is to retain the clarity on the surfaces between the different paints and the liquid to depict the features of flow streams precisely. Such clear surfaces are maintained with ox gall and water in real marbling. Several researchers have attempted to simulate this marbling effect with grid-based advection schemes. However, the Eulerian approach comes with built-in "numerical diffusion," which is notorious for blurring clear outlines. A higher-order accurate advection scheme can be used to prevent the dissipation, but it is computationally expensive and suffers from instabilities known as Gibb's oscillations. Among the computer graphics community, a great variety of fluid phenomena, such as smokes and liquids, have been explored [4]. However, these techniques are not directly applicable to marbling flow because they are carefully engineered to focus on producing specific types of visual properties.

Our method is related to an explicit surface tracking method, also known as front tracking, which is a technique for tracking propagating interfaces. Front tracking works with Lagrangian surface particles connected to triangles or piecewise linear curves, and it utilizes the underlying motion to capture deforming surfaces. Front tracking often outperforms other Lagrangian methods because the particles are placed only on the surface rather than filling the volume. However, the algorithm tends to be complicated because surfaces can be tangled.

In the proposed method, we track the deformation of the marbling shape with explicit surface particles based on the principle that the contour of the paint region rarely collides by advection due to the divergence free property of free surface fluid flow, as shown by Ando and Tsuruno [3]. Note that the ignorance of topology is only true for free surface flow in continuum level. In contrast to liquid animation, where the liquid domain merges or splits vividly over time, the liquid domain of free surface flow is usually fixed. This strategy makes the algorithm simpler and intuitive because the topological changes can basically be ignored. However, if no topological changes are taken into account, the number of vertices grows limitlessly as the contour stretches, which significantly slows the simulation. To permit the simulation of proportionately larger surfaces, we run a fundamental algorithm on the GPU in a parallel manner, and we resample the contour

* Corresponding author.
E-mail addresses: and@verygood.aid.design.kyushu-u.ac.jp (R. Ando), tsuruno@design.kyushu-u.ac.jp (R. Tsuruno).

0097-8493/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cag.2010.11.002

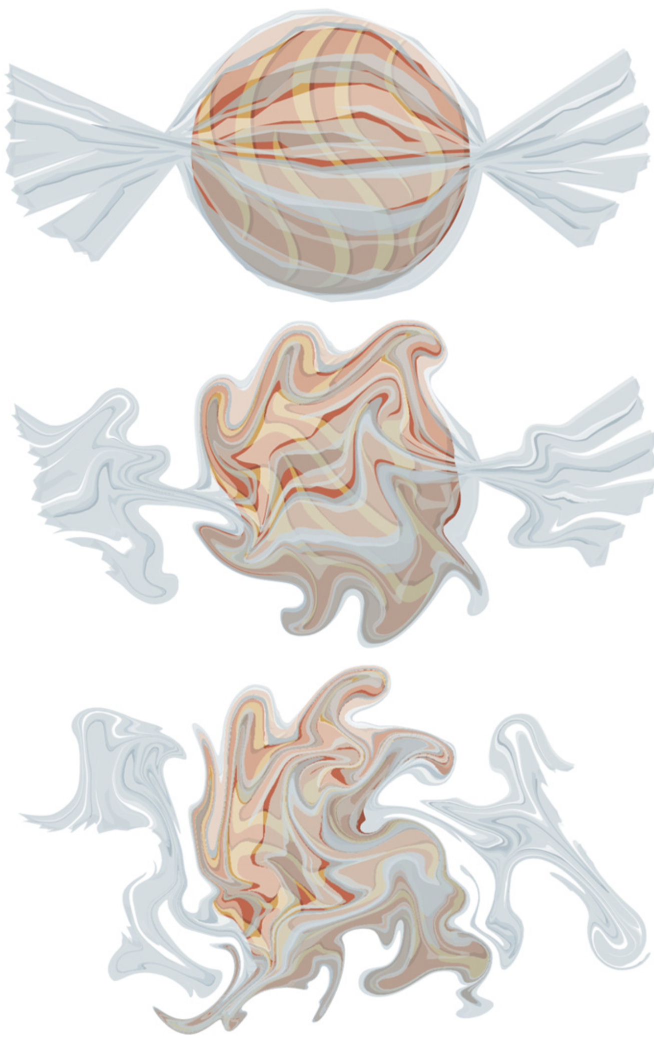

Fig. 1. *Marbleized candy clip*: This candy was deformed interactively using our marbling simulator, and it can be described in terms of vector graphics.

adaptively, watching for the local vorticity and curvature of the surfaces to approximate the shape with fewer vertices without losing much visual detail. Even though the contour is essentially collision free, because the surface is discretized over space and time, collisions can be produced due to numerical error. We found that this error does not produce significant visual artifacts, but as an option we also remove thin line regions that are almost invisible to remove a large number of vertices and collisions. Consequently, our algorithm runs reasonably fast in terms of shape complexity (Fig. 1). Our marbling simulation runs on an underlying velocity field of fluid, which is generated in real-time in response to the user's interactions. The characteristics of marbling deformation are controlled by the behavior of the velocity field. However, because the resolution of fluid flow is rather coarse, the tracker produces slight volume loss at each time step, which accumulates over time. To maintain concentration constant volume of the fluid, we slightly move surfaces in normal directions to effectively correct the error.

1.1. Simulation overview

For each time step, our marbling simulator takes the following five steps in order.

1. *Convection*: We start the simulation by generating the underlying fluid velocity with a uniform grid and semi-Lagrangian method. We advect the contour points explicitly by fetching velocities from 16 grid points using cubic spline interpolation with the fourth-order Runge–Kutta method. We also subdivide the stretched contour by rewinding time to find a more accurate subdivision point than that obtained with linear subdivision.
2. *Adaptive sampling*: We resample the contour points according to the local curvature, vorticity and distance from the opposite contour to depict the shape with fewer vertices.
3. *Volume error correction*: If we advect the marbling shape under the fluid motion with a coarse grid, the volume error accumulates slowly over time. We quickly correct this error by inflating or shrinking the entire shape toward normal directions with slight changes.
4. *Shape simplification*: We dynamically remove extremely thin lines that are nearly invisible so the simulator will run more smoothly. To do this, we detect and cut such regions, and then reconstruct the contour connections. The simplification test can be triggered at any time, particularly when the user desires, because the collision can basically be ignored.
5. *Rendering and export*: The deformed marbled shape is rendered through the graphics hardware or exported in an editable vector graphics format.

2. Previous work

Our work is related to two categories of studies: the artistic expression of fluids and surface tracking methods.

The most relevant field is the direct simulation of marbling. Acar and Boulanger [2] attempted to reproduce visual marbling effects using a physically derived flow model. They observed surface flow based on mesoscale dynamics, and they produced fluctuation effects at different scales. To advect clear silhouettes under Eulerian grids, they employed B-spline interpolation and extended the range of concentration temporarily in the semi-Lagrangian advection phase. However, Eulerian grid approximation is limited in terms of the degree of resolution if we wish to obtain a reasonable simulation. If we simulate with high resolution, a great deal of per-pixel computation and memory is required. Zhao et al. [33] developed a real-time marbling simulator that fully runs on the GPU. They employed a third-order accurate, but fast, unsplit semi-Lagrangian constrained interpolation profile method to reduce the numerical dissipation while retaining stability. Although they achieved 24 frames per second (FPS) at 1680×1050 grid resolution, the method was still inadequate for printing large materials because the resolution was approximately 5×3 square inches at 350 dpi. Acar [1] also proposed a level set-based system that provides a flexible environment for the user to generate traditional marbling patterns in high resolution. However, real-time feedback is still computationally expensive with this method.

Eden et al. [6] proposed a method for rendering liquids in a cartoon-style manner. By exploiting a physically created fluid surface, they rendered the effect by emphasizing the properties of the liquid's shape and motion, which were inspired by the abstraction and simplification of cartoon animations. This method resembles our own in that both have clear silhouettes and few colors. However, they used an implicit contouring method as the underlying liquid animation. Hence, the thin line detail is inherently smoothed out before it is stylized.

Selle et al. [28] introduced a technique for generating cartoon-style animations of smoke. Based on a physically based simulated output, they traced marker particles and rendered them using depth buffer differences to generate clear smoke animations. McGuire and Fein [19] extended this technique and developed a system for rendering real-time animations of smoke in addition to introducing a novel self-shadowing algorithm. However, these