Screen Space Fluid Rendering with Curvature Flow

Maarten Terpstra* and Bram Musters*

*Department of Computing Science
University of Groningen
Nijenborgh 9
Groningen 9747 AG
{m.l.terpstra, b.t.musters}@student.rug.nl

Abstract
This should contain a brief description of the paper, setting it in the context of other work, describing its main features and indicating the advances achieved. The abstract is meant to be succinct, not extending beyone about half a dozen lines, but should give potential readers an idea of the paper's potential importance.

Keywords:

1 Introduction

Fluids are abundant in nature: whether close to our homes, falling from the sky, deep in the earth in the form of magma, or in large volumes as lakes or oceans, fluids are nearly guaranteed to be found everywhere. These fluids are not only necessary for most animals roaming the planet to survive, but are also used by humans in industrial processes to obtain power, starch, sawed wood or cooling in our engines.

For some applications it may be useful to simulate these fluids using computers in order to improve usage of the fluids in certain situations. By using visualizations of fluids bottlenecks or missed opportunities in systems may be identified and the model can modified to improve the system. This kind of visualizations are mainly used in engineering.

Other types of visualizations, which are mainly used in academia, includes fields of research as oceanography, volcanology and astrophysics. In these kind of visualizations, entire systems are modeled to identify breaking points or how different parameters may affect an entire system.

There are several methods available for simulating fluids, such as the Eulerian method or the Smoothed Particle Hydrodynamics (SPH). The Eulerian method is fairly straightforward, as it describes fluid motion as an integration of the surface over time. This simple technique comes with a downside in the fact that it requires a (finite) grid on which the fluid moves, which is computationally expensive and limits the fluid in that it cannot move everywhere. SPH proves to be a viable alternative. With this technique, a body of fluid is reduced to a number of particles, where each particle has a position, velocity, acceleration and density. These properties can be used to extract a smoothed surface which can then be visualized in an aesthetically pleasing way.

Simulating a fluid is only half the battle. The insight gained from a fluid rendering can increased by having a natural visualization of the system.

2 Problem Definition

Since the fluids are simulated using SPH, the fluid is discretized in different particles. A scene containing a fluid, simulated by SPH, can be seen in figure 1. Since this "fluid" does not look like a fluid that can be found in nature, a visualization technique has to be applied.

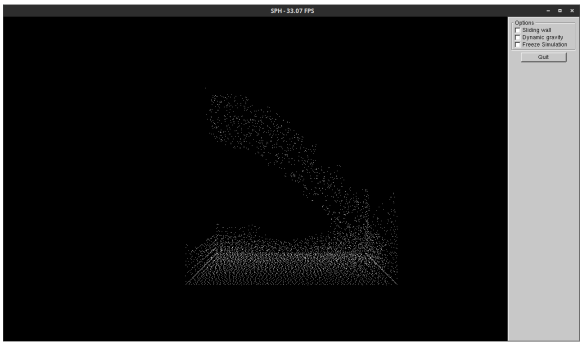

Figure 1: SPH simulation

It is our goal to create a nature-like rendering of fluids that can be rendered in real-time, in order to use it in games, for example. It is also desirable to make sure that the rendering can be customized according to the requirements of users. For example, fluids can have different kind of thicknesses, or the quality of the rendering can be altered in order to keep the rendering in real-time.

2.1 Related work

Various methods are developed that try to achieve a nature-like rendering of fluid. Some of these techniques require to use a mesh, which is not desirable. Other techniques have the drawback that they can not be rendered in real-time.

Zhang et al. [2008] developed a method that makes use of point-based rendering, therefore no grid is needed anymore. However, a drawback of this method is that it results in unreasonably thick surfaces.

Solution

The paper introduced by van der Laan et al. van der Laan et al. [2009] provides a fitting solution to our problem. The paper describes a method for visualizing fluids simulated using SPH in a natural way. The goal of the paper is to provide a solution to our aforementioned problem. According to the paper, they want to create:

1. Achieves real-time performance, with a configurable speed versus quality trade-off.
2. Does all the processing, rendering and shading steps directly on the graphics hardware.
3. Smoothes the surface to prevent the fluid from looking blobby or jelly-like.
4. Is not based on polygonization, and thus does not suffer from the associated tesellation artifacts.

Items 1 and 3 have a direct connection with our goals and items 2 and 4 are ways to enable these goals.-

3 Implementation

The solution that has been proposed in the previous section is implemented using the C++ programming language in combination with CG as shading language. The actual SPH fluid simulation is given and can be seen in figure 1, by implementing the multiple passes described in the previous section, a nature like fluid is expected.

3.1 Depth determination

The depth at each pixel of each particle closest to the camera is determined using a combination of a vertex- and fragment shader. Each particle has a position in world space that is passed to the vertex shader. These particles are rendered as spheres to determine the correct depth values.

The vertex shader computes and passes the following properties of each particle to the fragment shader:

- Position of center in eye space.
- Position of center in screen space.
- Splat size.
- Splat radius.

The fragment shader uses this input to determine the depth at every fragment. To determine this depth, the splat is rendered as a sphere by discarding fragments that fall outside the sphere. From this, the normal from the center of the sphere towards its surface is determined by taking the difference of the current fragment position and the current particle center.

Results

References

van der Laan, W. J., Green, S., and Sainz, M. (2009). Screen space fluid rendering with curvature flow. In *Proceedings of the 2009 symposium on Interactive 3D graphics and games*, 91–98. ACM.

Zhang, Y., Solenthaler, B., and Pajarola, R. (2008). Adaptive sampling and rendering of fluids on the gpu. In *Proceedings of the Fifth Eurographics/IEEE VGTC conference on Point-Based Graphics*, 137–146. Eurographics Association.