Real Time Multiple Fluid Simulation and Rendering on GPUs

Dunfan Lu

Somerville College

University of Oxford

Supervised by: Joe Pitt-Francis

3rd Year Project Report for

MCompSci

Trinity 2020

Abstract

Fluid simulation is an extremely common and important computational task. These simulations are often heavily expensive and require a large amount of CPU time, hence difficult to apply in real time applications. Fortunately, modern GPUs, equipped with massively parallel general purpose computing architectures, provide a solution to this problem. This project explores methods to perform fluid simulations on GPUs, and to realistically render the simulated fluids, both in real time.

This project focuses on three of the most widely used fluid simulation algorithm: FLIP (Fluid-Implicit-Particle), PBF (Position Based Fluids), and PCISPH (Predicative-Corrective Incompressible Smoothed Particle Hydrodynamics). The project studies how each algorithm can be parallelized, and creates efficient GPU implementations of these algorithms using NVIDIA's CUDA programming model. The project also extends the FLIP algorithm to support multiple fluid rendering, thereby capturing the diffusion phenomenon between different phases of fluids. Alongside the simulation, the project implements a real time liquid rendering scheme, which efficiently captures the refection and refraction effects, as well as the varying concentrations of colored fluid phases inside the liquid. These algorithms are integrated into a fully functional program, which accepts user-provided simulation parameters, then performs and renders the liquid simulation in a real time and visually plausible manner.

Contents

1 Introduction

1.1 Motivation

Fluids can be seen everywhere. The smoke coming out of chimney spreading in the wind, the milk in a cup mixing with the coffee, the calm flow of a river with tiny ripples under the rain, and the enormous waves of the ocean splashing onto the surface. Many of these these phenomenons have interesting and even stunning visual effects, thus, quite often, plausible images of these fluids need to be computationally generated for purposes such as cinematics and video gaming.

Due to the complexity underlying the behavior of fluids, accurate numerical simulation of fluids often require a huge amount of computational resources. In areas such as aeronautic engineering, where the importance of accuracy completely overrides that of efficiency, hours of CPU time can be spent on the simulation of a few seconds of fluid motion. Real time computer graphics applications, such as video games, usually do not require this level of accuracy, but instead asks the simulation to be computed in roughly the same amount of time as the physical process it represents. This efficiency requirement is met with the help of modern GPUs, which have massively parallel computing abilities. This project demonstrates this by implementing three simulation algorithms on GPUs, all of which can perform simulations in real time.

For computer graphics applications, fluid simulation isn't the only task. It is equally necessary for the software to display the results of simulation (i.e, the shape and motion of the fluid) to the user. This is especially important for interactive video games, where the user experience highly depends on the quality of the rendering, maybe even more so than the accuracy of simulation. This project thus studies and implements the real time photorealistic rendering of a special, but likely most important, type of fluid: liquids.

1.2 Related Work

The study of the behavior of fluids dates back to 18th century, when Leonhard Euler proposed a set of partial differential equations (PDEs), known as *Euler Equations*, that governs the behavior of an idealized incompressible and inviscid fluid. In the 19th century, these equations were extended by Claude-Louis Navier and George Gabriel Stokes into the famous *Navier-Stokes Equations*, which describe a much wider class of fluids that occur in the real world. These equations are exactly what most fluid simulation softwares, including the one implemented in this project, are trying to solve.

Somewhat unfortunately, the Euler and Navier-Stokes equations have extremely difficult mathematical properties, and general analytical solutions are yet to be found even today. As a result, softwares resort to numerical methods to approximate solutions. In computer graphics applications, there are two main families of numerical methods for solving the fluid equations: the grid-based methods and the particle-based methods. Each approach comes with its own benefits and drawbacks, but could both be implemented efficiently on GPUs to achieve real time simulation.

The grid-based methods relies on spatial discretizations of the fields (e.g the velocity field) that represents the fluids. The most widely used discretization method, known as the **MAC** (Marker and Cell) **grid**, was proposed by Harlow and Welch [2] in 1965. This grid offers second order accuracy, and is used as a basis of most fluid simulation algorithms.

A significantly important step during a grid-based simulation is to move all the physical quantities stored in the grid (e.g velocity, concentration) according to the velocity field. This step, known as **advection**, essentially determines how the fluid moves and thus how its shape changes, and is key to the quality of a simulation. A few popular advection algorithms include **MacCormack**[8] and **BFECC**[4], both of which have efficient GPU implementations[1][10]. The most widely used advection algorithm is known as **FLIP** (Fluid Implicit Particle)[11], developed by Zhu and Bridson. This algorithm, interestingly enough, makes uses of particles to move quantities within the MAC grid. FLIP has various advantages over the purely grid-based algorithms, and this project demonstrates how the original FLIP can be efficiently adapted to work on GPUs.

As an addition to the traditional single phase fluid simulation, Kang et el.[3] showed how to extend the grid-based algorithms to capture the diffusion between

multiple fluid phases (e.g red ink diffusing in transparent water). This project implements a modified version of the proposed algorithm, where FLIP, rather than BFECC, is used to advect the concentration of different fluid phases.

Parallel to the grid-based approach is the family of particle-based algorithms. For computer graphics, the most commonly used class of particle-based algorithms is known as **SPH** (Smoothed Particle Hydrodynamics). Introduced to computer graphics in 2003 by Müller [7], the SPH method represent the fluid by a moving cloud of particles, which carry the physical quantities with them. This project chooses to implement two extensions to SPH: **PCISPH**(Predicative-Corrective Incompressible SPH) by Solenthalar[9] and **PBF**(Position Based Fluids) by Macklin and Müller[6]. These extended algorithms improve upon the plain SPH in that they enforce the incompressibility constraint of fluids, which are important for visual fidelity.

Given a well performing simulation, either grid-based or particle-based, it remains a nontrivial task to visualize the fluid. This project follows the proposal by Zhu and Bridson [11], who showed how a particle representation of a fluid can be used to compute a signed distance field, which represents the distance to the fluid surface of each point in the 3D space. An algorithm known as Marching Cubes [5], proposed by Lorensen, can then use this field to reconstruct the surface of the fluid into a triangle mesh representation, which is suitable for rendering.

1.3 Outline of Project and Report

This project focuses on investigating and producing highly performant GPU implementations of the most widely used fluid simulation and rendering algorithms. An extended version of the FLIP algorithm, which supports multiple fluid simulation and diffusions between fluids of different colors, is studied and implemented, with the details elaborated in chapter 3. Similarly, GPU versions of the PCISPH and PBF algorithm are also created, as described in chapter 4.

To visualize the simulations, the project implements a fast surface reconstruction algorithm, which transforms a particle cloud representation of fluids into a renderable triangle mesh. A real time renderer is implemented to render the mesh while capturing all the reflection and refraction phenomenons that occur in the real world. Furthermore, the renderer takes into account the different levels of attenuation of light caused by fluids of different colors, thereby also realistically rendering the liquid diffusion effects. The details of the renderer are given in chapter 5.

These implementations are based from their origin descriptions in the papers, but many additional considerations and optimizations were taken to enable efficient parallelization. Specifically, the project utilizes NVIDIA's general purpose GPU programming interface known as CUDA, and tailors the implementation code to exploit the full potential of CUDA GPUs. The results are showcased by a fully functional program, which allows the user to easily configure the starting state of a simulation. These include the shapes and sizes of the fluid before the simulation starts, as well as the initial color and transparency of each fluid volume. The program can then carry out the simulation and render realistic results to the user in real time.

2 Physics of Fluids

2.1 Vector Calculus

2.2 The Incompressible Euler and Navier-Stokes Equations

$$\begin{cases} \dfrac{D\mathbf{u}}{Dt} = -\dfrac{\nabla p}{\rho} + \mathbf{g} \\ \nabla \cdot \mathbf{u} = 0 \end{cases} \qquad \text{(Euler Equations)}$$

$$\begin{cases} \dfrac{D\mathbf{u}}{Dt} = -\dfrac{\nabla p}{\rho} + \mathbf{g} + \nu \nabla \cdot \nabla \mathbf{u} \\ \nabla \cdot \mathbf{u} = 0 \end{cases} \qquad \text{(Navier-Stokes Equations)}$$

3 Grid-Based Simulations

4 Particle-Based Simulations

5 Rendering

A Sample Title

B Sample Title

Bibliography

[1] Nuttapong Chentanez and Matthias Müller. Real-time eulerian water simulation using a restricted tall cell grid. In *ACM Siggraph 2011 Papers*, pages 1–10. 2011.

[2] Francis H Harlow and J Eddie Welch. Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. *The physics of fluids*, 8(12):2182–2189, 1965.

[3] Nahyup Kang, Jinho Park, Junyong Noh, and Sung Yong Shin. A hybrid approach to multiple fluid simulation using volume fractions. In *Computer Graphics Forum*, volume 29, pages 685–694. Wiley Online Library, 2010.

[4] ByungMoon Kim, Yingjie Liu, Ignacio Llamas, and Jaroslaw R Rossignac. Flowfixer: Using bfecc for fluid simulation. Technical report, Georgia Institute of Technology, 2005.

[5] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. *ACM siggraph computer graphics*, 21(4):163–169, 1987.

[6] Miles Macklin and Matthias Müller. Position based fluids. *ACM Transactions on Graphics (TOG)*, 32(4):1–12, 2013.

[7] Matthias Müller, David Charypar, and Markus Gross. Particle-based fluid simulation for interactive applications. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 154–159. Eurographics Association, 2003.

[8] Andrew Selle, Ronald Fedkiw, Byungmoon Kim, Yingjie Liu, and Jarek Rossignac. An unconditionally stable maccormack method. *Journal of Scientific Computing*, 35(2-3):350–371, 2008.

[9] Barbara Solenthaler and Renato Pajarola. Predictive-corrective incompressible sph. In *ACM SIGGRAPH 2009 papers*, pages 1–6. 2009.

[10] Shibiao Xu, Xing Mei, Weiming Dong, Zhiyi Zhang, and Xiaopeng Zhang. Interactive visual simulation of dynamic ink diffusion effects. In *Proceedings of the 10th International Conference on Virtual Reality Continuum and Its Applications in Industry*, pages 109–116, 2011.

[11] Yongning Zhu and Robert Bridson. Animating sand as a fluid. *ACM Transactions on Graphics (TOG)*, 24(3):965–972, 2005.