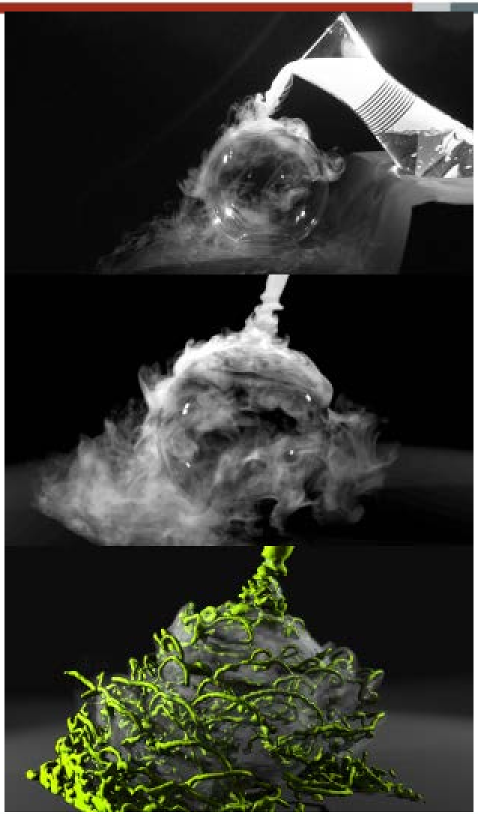
St Petersburg University
www.spbu.ru

GPGPU implementation of Schrödinger's Smoke for Unity3D

Oleg Iakushkin, Anastasia Iashnikova, Olga Sedova

Introduction



In this work describes an algorithm for Eulerian simulation of incompressible fluid - Schrödinger's Smoke.

Schrödinger's Smoke can robust simulate complex phenomena, for example, interacting vortex filaments, even on grids with small dimensions.

Main Idea

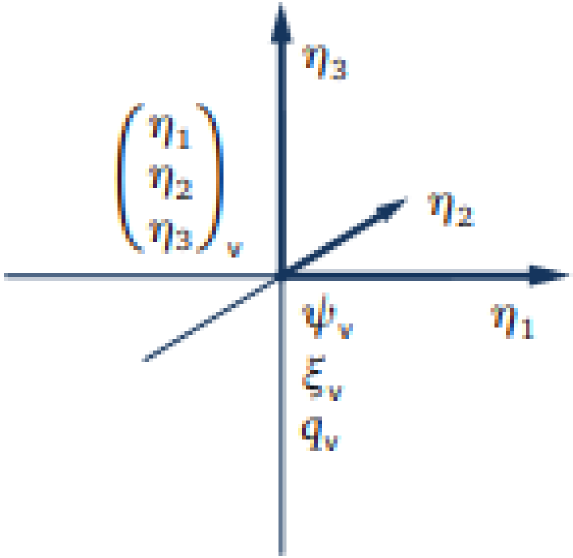
Simulations are performed on a 3D lattice with vertex set.

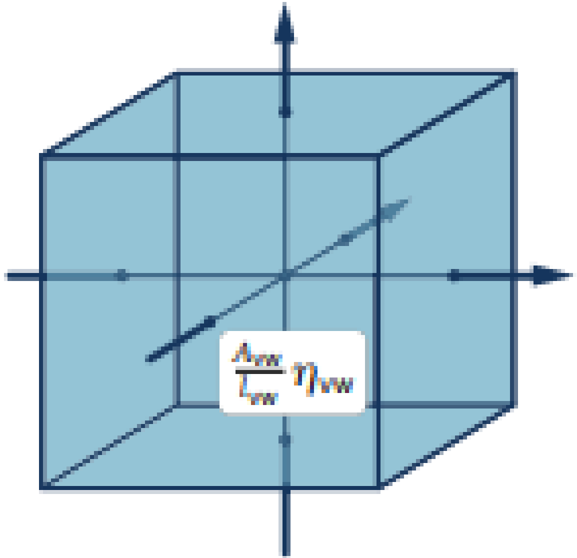
Vertices need to store samples of the wave function ψ_v, the real-valued pressure q_v and the real-valued divergence ξ_v.

The discrete velocity 1-form is defined on directed edges and stored in staggered grid fashion at vertices.

The discrete divergence is the usual signed sum over incident edges, weighted by the quotient of dual facet area A_{vw} to edge length l_{vw} and normalized by dual cell volume V_v following standard Discrete Exterior Calculus conventions.

Presentation of the main idea





Realization

The algorithm is based on representing models as a system of particles.

Each particle represents a small portion of a fluid or amorphous material. A particle has a certain 'lifespan', during which it may undergo various changes.

CUDA Implementation and Rendering of Schrodinger's Smoke
by Yixiu Zhao (yixiuz) and Shangda Li (shangdal)

→ 0-10% Speedup ☹

Basic Algorithm

Algorithm 1 Basic ISF

Input: $\psi^{(0)}, dt, \hbar$ ▷ Initial state and parameters

1: **for** $j \leftarrow 0, 1, 2, \ldots$ **do**
2: $\psi^{\text{tmp}} \leftarrow \textsc{Schrödinger}(\psi^{(j)}, dt, \hbar)$
3: $\psi^{\text{tmp}} \leftarrow \psi^{\text{tmp}} / |\psi^{\text{tmp}}|$ ▷ Normalization
4: $\psi^{(j+1)} \leftarrow \textsc{PressureProject}(\psi^{\text{tmp}})$
5: **end for**

Basic Equation

Algorithm 2 Time integration of Schrödinger equation

1: **function** $\textsc{Schrödinger}(\psi, dt, \hbar)$
2: $\hat{\psi} \leftarrow \text{FFT3D}(\psi)$
3: $\hat{\psi} \leftarrow e^{i\lambda dt \frac{\hbar}{2}} \hat{\psi}$
4: **return** $\textsc{InvFFT3D}(\hat{\psi})$
5: **end function**

Time Step

Basic Algorithm

Pressure Project

Algorithm 3 Divergence free constraint

1: **function** PressureProject(ψ)
2: **for each** vw $\in \mathcal{E}$ **do** ▷ Scaled velocity 1-form at edges
3: $\tilde{\eta}_{vw} = \arg\langle\psi_v, \psi_w\rangle_{\mathbb{C}}$ ▷ $\hbar^{-1}$ multiple of Eq. (4)
4: **end for**
5: **for each** v $\in \mathcal{V}$ **do** ▷ Scaled divergence at vertices
6: $\xi_v = \frac{1}{V_v}\sum_{vw\in\mathcal{E}} \frac{A_{vw}}{l_{vw}}\tilde{\eta}_{vw}$ ▷ Eq. (5)
7: **end for**
8: $\hat{\xi} \leftarrow$ FFT3D(ξ)
9: $\hat{\xi} \leftarrow \hat{\xi}\begin{cases}\tilde{\lambda}^{-1} & \text{if } \tilde{\lambda} \neq 0 \\ 0 & \end{cases}$
10: $q \leftarrow$ InvFFT3D($\hat{\xi}$)
11: **return** $e^{-iq}\psi$
12: **end function**

Problem Statement

- Slow work
    - (runs on the CPU)

Programming language

- (requires knowledge of the language for the job)

Implementation for Unity3D

The computer algorithm "Schrodinger Smoke" was moved and implemented in the development environment of Unity3D, with the used ArrayFire library which provides OpenCL and CUDA.

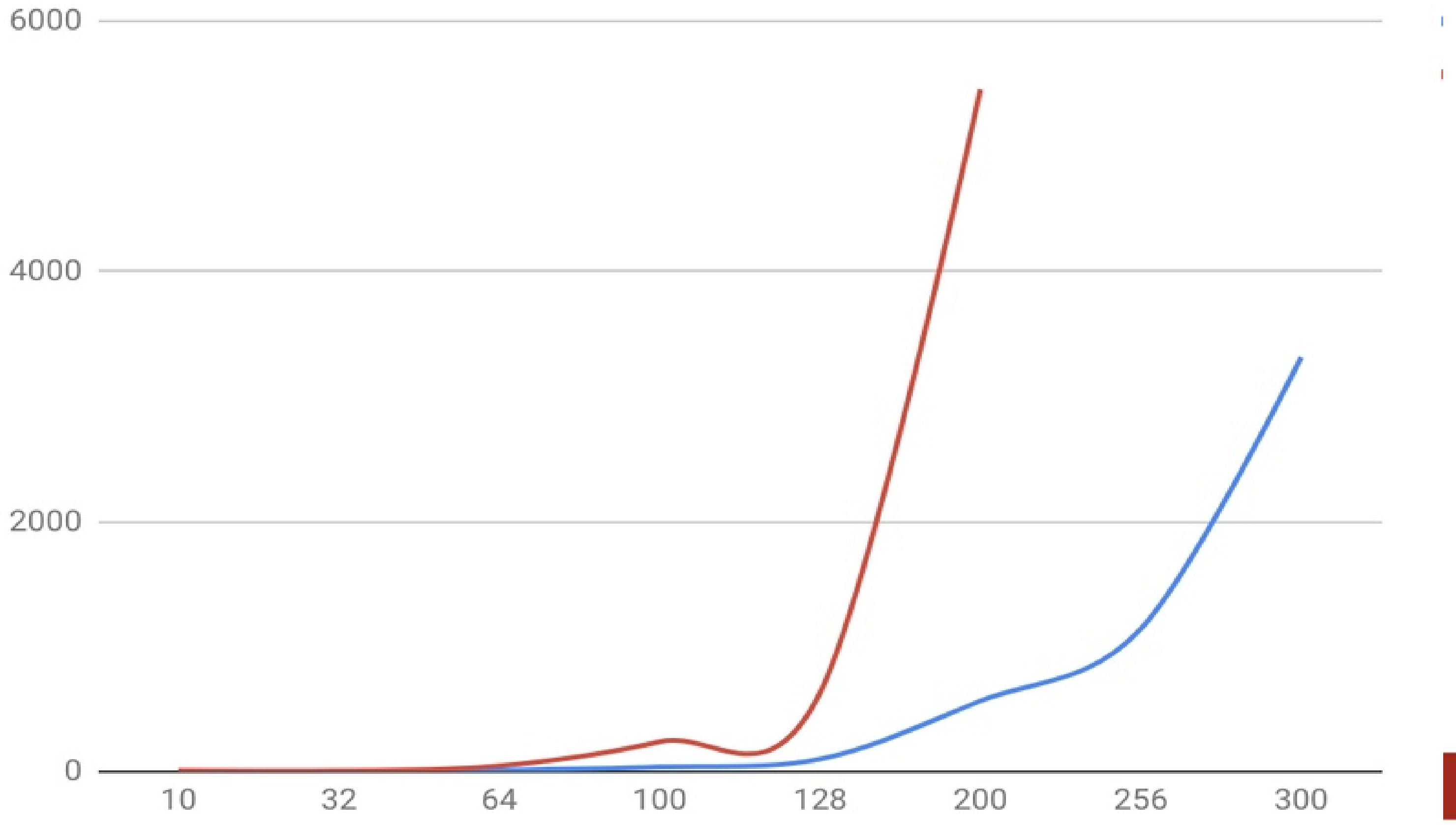
This allowed to transfer most of the computational load of the calculations of physical processes to GPGPU.

C# and GPU

ArrayFire-dotnet was used to port Time Step and Pressure Project functions to GPGPU.

ArrayFire supports OpenCL, CUDA, and CPU execution => can be used in Crossplatform Simulations.

Evaluation



THANK YOU FOR ATTENTION!

Speaker: Oleg Jakushkin