Algorithmic Fluid Imagery

Mark J. Stock
115 Langley Rd
Newton, MA 02459 USA
mstock@umich.edu

Abstract

The capability of computers and algorithms to solve once-intractable problems arising in physics gives new media artists powerful tools for communication and creation. In this paper, we will present two common methods for simulating a variety of physical systems: one based on discretizations of field values onto cells, and the other onto particles. We will draw parallels between these and much simpler Cellular Automata, and give many examples of work using a variety of methods. Although the emphasis will be on fluid dynamics, the equations and algorithms summarized herein are applicable to many fields of physics.

1 Image-Making With Interacting Automata

The simplest way to use mathematics and algorithms to generate imagery is to have each pixel's color determined by an independent calculation (i.e. not influenced by its neighbors or previous calculations). Most raytracing and rendering methods and all fractal and iterated function system (IFS) renderers do this. In contrast, schemes that use interacting automata (IA) are "grown" from multiple iterations of images, each involving limited or widespread interaction between pixels or computational elements. IA simulations can require very simple interaction rules but nevertheless be very expressive, capable of infinite detail and showcasing emergent behavior. Though the simplest images are defined via pixels, most of the methods presented below are capable of creating both rasterized and vectorial imagery.

Cellular Automata (CA). Cellular automata are the simplest forms of interacting automata. In CA, properties in cells in a regular matrix are updated using information from adjacent cells and/or preceding time steps. Conway's *Game of Life* [6] is a CA played on a two-dimensional grid. In the standard Game of Life, a cell is "born" if it has exactly 3 neighbors, stays alive if it has 2 or 3 living neighbors, and dies otherwise. Despite the simplicity of these rules, Life generates a remarkable diversity of behaviors, and is still the subject of active research forty years after its creation. Another well-studied simple CA is Wolfram's "Rule 110" [11]. In this, each bit in a one-dimensional sequence is evolved in pseudo-time using information about itself and its two neighbors from the previous step. Starting from one "on" bit (at the top) and working down one row at a time generates the first image in Figure 1. Unlike fractals, a single bit in this image can only be calculated by running the simulation in reverse along an upside-down pyramid through pseudo-time (a feature not common to some CA).

Elements in motion. When one of the variables in an element's description is its location, the automata are considered mobile. Differential movement between elements means that an automaton's neighbors change in time and space. This additional level of complexity gives these still-simple systems much greater ability to recreate natural imagery. Even when constrained to move on a regular grid, IA simulations with moving automata can generate detail-filled algorithmic imagery; Resnick's termites [9] and diffusion-limited aggregation (DLA) [1] are two ideal examples of this.

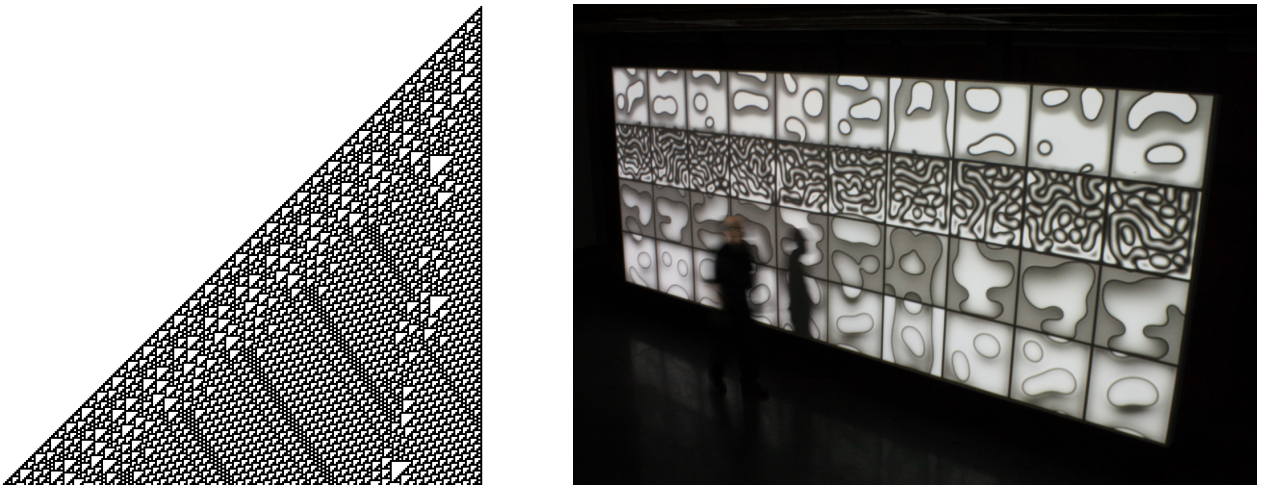

Figure 1 : *(Left) Wolfram's Rule 110; (right) reaction-diffusion textures in Brian Knep's* Drift Wall *(2007).*

2 Computational Fluid Dynamics (CFD)

Computational Fluid Dynamics concerns the numerical simulation of the motion of fluids on computers. A special case of CA, the specific rules used in CFD for interaction among neighboring cells are generated from discrete approximations to the differential equations that describe the underlying physics. Unlike CA, CFD schemes allow real-valued state variables, and are thus examples of Coupled Map Lattice systems.

One-dimensional diffusion. A simple example of using CA-like rules to simulate a physical system is that of a finite-difference approximation to the unsteady one-dimensional partial differential equation (PDE) for diffusion

$$\frac{\partial \zeta}{dt} = \alpha \nabla^2 \zeta = \alpha \frac{d^2 \zeta}{dx^2} \tag{1}$$

where α is the diffusion coefficient for scalar ζ, and ∇^2 is the Laplace operator. Think of ζ as being the temperature profile along a thin bar or the species fraction in a tube. The diffusion equation is used frequently in simulations of reaction-diffusion textures, and has been used expressively by artist Brian Knep (Figure 1).

To facilitate its numerical solution, time and space are discretized into uniform segments of lengths Δt and Δx, respectively. A Taylor-series expansion of the time derivative generates the forward integration step (Euler's method)

$$\zeta_{(t+\Delta t)} = \zeta_{(t)} + \Delta t \left. \frac{\partial \zeta}{\partial t} \right|_t + \mathscr{O}(\Delta t^2) \tag{2}$$

and a second-order central-difference operator approximates the spatial derivative

$$\frac{d^2 \zeta}{dx^2} = \frac{1}{\Delta x^2} \left(\zeta_{(x-\Delta x)} - 2\zeta_{(x)} + \zeta_{(x+\Delta x)} \right) + \mathscr{O}(\Delta x^2). \tag{3}$$

Combining these and dropping the higher-order terms generates an equation for the approximate values of ζ at the subsequent time step as a function of the values from the current step.

$$\zeta_{(x,t+\Delta t)} = \zeta_{(x,t)} + \alpha \frac{\Delta t}{\Delta x^2} \left(\zeta_{(x-\Delta x,t)} - 2\zeta_{(x,t)} + \zeta_{(x+\Delta x,t)} \right) \tag{4}$$

Going further and setting $\Delta t = \frac{\Delta x^2}{4\alpha}$ gives

$$\zeta_{(x,t+\Delta t)} = \frac{1}{4} \left(\zeta_{(x-\Delta x,t)} + 2\zeta_{(x,t)} + \zeta_{(x+\Delta x,t)} \right). \tag{5}$$

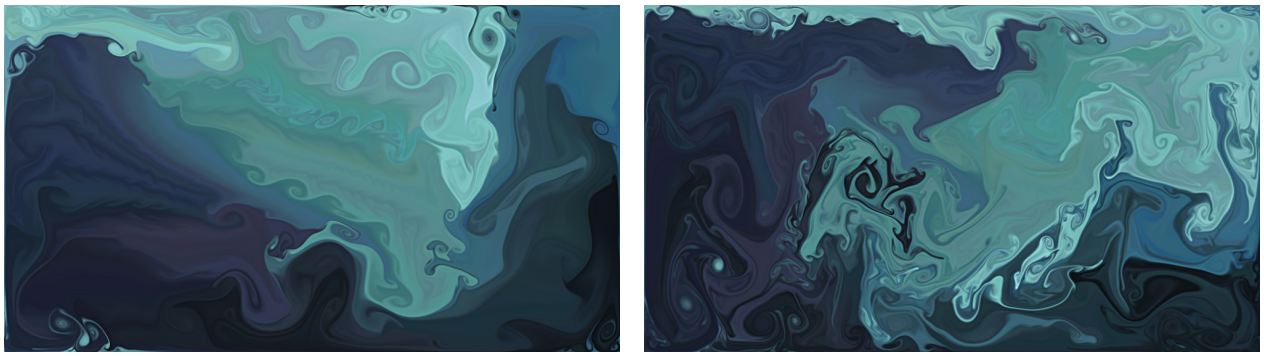

Figure 2 : *Two frames from* Gyre 29 *(2011), made with a traditional grid-based finite difference solver using semi-Lagrangian advection.*

This simply sets the new cell's value to a weighted average of itself and its neighboring pixels, in much the same manner as the 1D cellular automata of the previous section.

The result of a simulation using this method with the initial condition of a δ-function peak of ζ is a gradually flattening Gaussian curve (a fundamental solution to the diffusion equation). This is of completely different character than Conway's Game of Life or Rule 110, owing to the averaging nature of the construction rule. Nevertheless, the discrete diffusion equation illustrates the underlying method for solving seemingly-intractable partial differential equations on numerical computers.

Navier-Stokes equations. Independently discovered around 1822 by Navier and Stokes, these differential conservation equations (one for mass, others for momentum, and an energy equation omitted for simplicity) describe the motion of continuum fluids. In a simplified notation, they are

$$\frac{\partial(\rho\mathbf{u})}{dt} + \mathbf{u}\cdot\nabla(\rho\mathbf{u}) = 0 \tag{6}$$

$$\rho\left(\frac{\partial\mathbf{u}}{dt} + \mathbf{u}\cdot\nabla\mathbf{u}\right) = -\nabla p + \nu\nabla^2\mathbf{u} \tag{7}$$

where ρ is density, $\mathbf{u}$ is the velocity vector, p is pressure, ν is kinematic viscosity, and del (∇) is the vector differential operator. Note how equation (7) is like equation (1) with $\mathbf{u}$ replacing ζ and two extra terms.

Analytic solutions to the Navier-Stokes equations only exist for a few special—and simple—cases, so usually solutions are approximated numerically. Like above, the fluid domain is discretized into an array of individual fluid cells (each with its own unique values for $\mathbf{u}$, ρ, and p) and time is advanced for the whole domain at once. These cells can be arranged in regular (square, rectangular, curvilinear), irregular (tetrahedra, other packed polytopes), or hierarchical (adaptive mesh resolution) patterns. Because of the many possible variations, the discretized version of equations (6-7) will be omitted. It is sufficient to mention that the form of the equations and the details of the discretization play an outsized role in the applicability and numerical properties of the resulting method [7]. Readers interested in pursing algorithmic image creation using these methods, with less emphasis on realism, may consult Bridson [4].

A simulation representative of the capabilities of this method appear in Figure 2. In this case, multiple layers of colored fluid with differing densities begin at an angle to gravity. As time progresses, irreversible computations smoothly evolve the colors to new configurations. Vortices and eddys, large and small, populate the images. Interestingly, though fractal relationships are used nowhere in this algorithm and do not appear in the Navier-Stokes equations, the fully-turbulent state of a fluid system exhibits fractal-like statistical similarity over multiple spatial scales [8].

3 Vortex Methods

Just as with general interacting automata methods, CFD methods can be categorized into methods based on their discretization of the underlying fields. Some discretize space, allowing material to flow among fixed cells, while others discretize the fluid itself, using convecting particles or elements to carry fluid properties. The difference is similar to that between raster and vector graphics. Vortex methods are the latter type, and the main property that is convected is related to the local rotation of the fluid. Vortex flows are not the only physical systems that can be described in this manner, so for the benefit of exposition, other simpler systems will be presented first.

Green's function solutions to Poisson's equation. The Laplace and Poisson equations are simple elliptical PDEs that appear in many fields of physics including heat diffusion, fluid dynamics, gravitation, electrostatics, and magnetostatics. Green's function is a fundamental solution to these equations and presents a useful and natural method for solving many problems in physics.

In Newton's law of universal gravitation, the acceleration on a body is equal to the gradient of the gravitational potential field (ϕ_g) according to

$$\mathbf{a} = \frac{\mathbf{f}}{m} = -\nabla\phi_g. \tag{8}$$

The gravitational potential field appears in a Poisson equation with scalar mass density field ρ as

$$\nabla^2\phi_g = 4\pi G\rho. \tag{9}$$

This can be solved via free-space Green's function solutions using point masses to obtain

$$\phi_g(\mathbf{x}) = -G\sum_j \frac{m_j}{|\mathbf{x}_j - \mathbf{x}|} \tag{10}$$

where G is the gravitational constant. Naturally, to conduct a simulation of the motion of point masses, equations (8) and (10) can be combined to generate the familiar formula for the gravitational acceleration of mass i

$$\mathbf{a}_i = \frac{\mathbf{f}_i}{m_i} = G\sum_j \frac{m_j(\mathbf{x}_j - \mathbf{x}_i)}{|\mathbf{x}_j - \mathbf{x}_i|^3}. \tag{11}$$

Scientists use this equation (and many algorithmic performance tricks) to simulate the evolution of galaxies and solar systems in what are called *N-body* methods. The left side of Figure 3 shows the traces of several point masses from a three-dimensional N-body simulation.

In a similar fashion, both motion and forces in electrostatic fields and magnetic fields can be represented by elemental interactions between participating particles. Xu and Mould [12] use this method to draw lines of constantly-changing curvature through magnetic fields. It should be no surprise that natural forms emerge, as we shall see the computation is very similar to that of vortical motion in fluid dynamics.

Vortices. A vortex, in the pure fluid dynamic sense, is a rotating mass of fluid, and maintains no special motion toward or away from its center. It is fortunate that many flows at human space and time scales are of low Mach number (compressibility effects are small, i.e. no shock waves) and high Reynolds number (inertial effects dominate viscous effects) and thus can be represented more efficiently as collections of vortices than as fluid cells of velocity. In this section we derive the equations for CFD methods based on vortices and show some images from works made using those methods.

Taking the curl ($\nabla\times$) of the Navier-Stokes equations (6-7), assuming incompressibility ($\nabla\cdot\mathbf{u} = 0$), replacing the left-hand-side with the substantive derivative

$$\frac{D}{Dt} = \frac{\partial}{dt} + \mathbf{u}\cdot\nabla \tag{12}$$

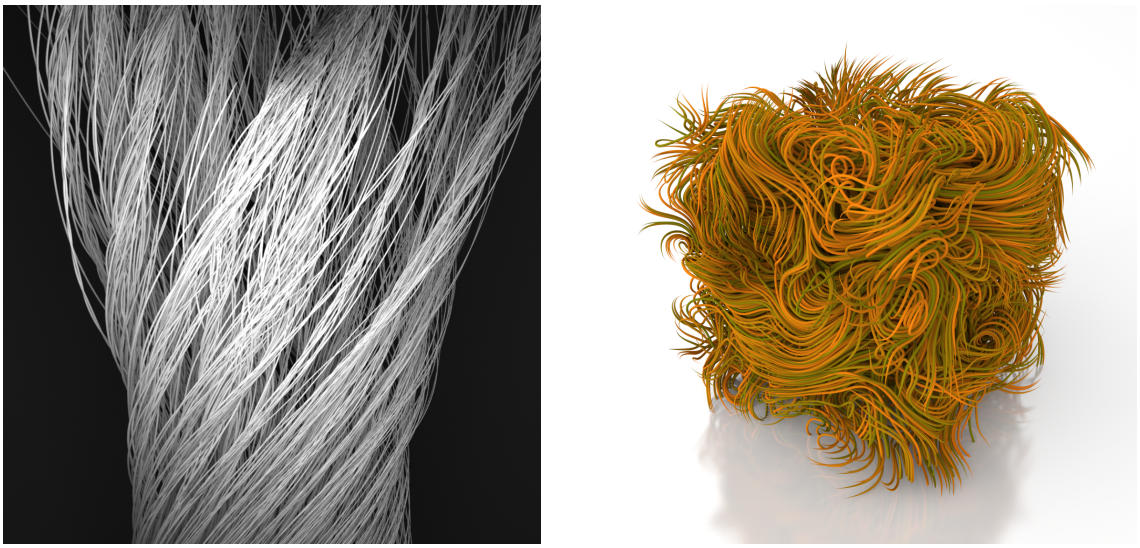

Figure 3: *(Left) particle paths from gravitational simulation (2005); (right)* Red Streamlines *(2004) depicting fluid traces through the curl of a vector field.*

(which tracks the change of a quantity as the observer convects with the medium), and introducing the quantity *vorticity*, defined as

$$\omega = \nabla \times \mathbf{u} \tag{13}$$

gives the vorticity transport equation

$$\frac{D\omega}{dt} = \omega \cdot \nabla \mathbf{u} + \nu \nabla^2 \omega. \tag{14}$$

The conservation of mass equation (6) is solved identically, and equation (14) does not contain the pressure term, which frequently causes numerical issues in velocity-based formulations. What remains is the convective derivative for vorticity, the "vortex stretching" term, and diffusion. In two dimensions the stretching term drops out, leaving only

$$\frac{D\omega}{dt} = \nu \nabla^2 \omega. \tag{15}$$

The nodes defining the element discretization (point, segment, or triangle), must convect through space, and to do that, the velocity at each node must be computed. Our formulation uses the *vector potential* ψ, and allows solution methods very similar to the gravitational and electrostatic problems from the previous section. Defining velocity as the curl of this vector potential

$$\mathbf{u} = \nabla \times \psi \tag{16}$$

we can write the equation for the vector potential using the vorticity field

$$\nabla^2 \psi = -\omega. \tag{17}$$

This can be solved, as before, via free-space Green's function solutions, resulting in the velocity induction law (also called the Biot-Savart law)

$$\mathbf{u}_i = \frac{1}{4\pi} \sum_j \frac{\Gamma_j \times (\mathbf{x}_j - \mathbf{x}_i)}{|\mathbf{x}_j - \mathbf{x}_i|^3} \quad \text{in 3D, and} \quad \mathbf{u}_i = \frac{1}{2\pi} \sum_j \frac{\Gamma_j \times (\mathbf{x}_j - \mathbf{x}_i)}{|\mathbf{x}_j - \mathbf{x}_i|^2} \quad \text{in 2D.} \tag{18}$$

These are summations over vortex elements defined as δ-functions of vorticity with finite *circulation* Γ. Note that circulation is a vector quantity in three dimensions, but a scalar quantity in 2D. As you can see,

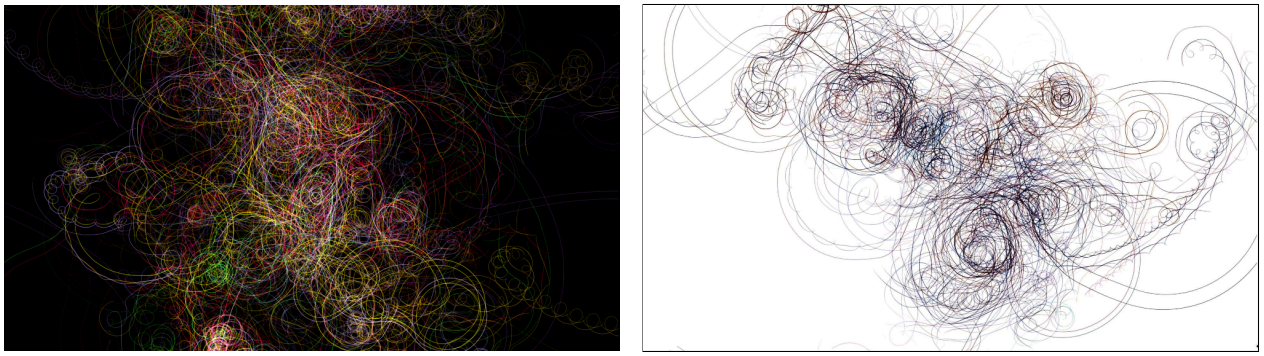

Figure 4 : *Frames from* Rota 9_5 *(2010) and* Rota 9_8 *(2010), examples of 2D vortex particle methods.*

there are only minor differences between this equation and equation (11), primarily the cross-product in the numerator. As such, algorithms suitable for one of these situations will work for the others with minor modification. These are the equations used to create the flowing curves in the image in Figure 3.

A vortex method is a CFD method that simulates fluid flow by tracking the positions and strengths of a series of vortex elements as they change in time using equations (14-15) and (18). Summaries of the history and research of vortex methods and numerous references are provided by Cottet and Koumoutsakos [5] and the author [10].

Taking artistic liberty with these equations opens up an interesting space for exploration while retaining the flow-like nature of the resulting dynamics. The images in Figure 4 are from two simulations of a very small number (85) of vortex-like particles. In both, particles are given random initial positions ($\mathbf{x}_i$) and circulations (Γ_i). Equation (18) is used to move the particles around with no change to their strengths. The particles trace thin lines on the background as they move, revealing their wispy, dance-like motions as time progresses. Owing to their simplicity, these simulations can be run in real-time on a laptop or desktop computer. Like gravitational simulations, the code required to create these images is short and can be taught to novice programmers in less than an hour.

Vortex lines and sheets. While most vortex methods discretize the vorticity over individual disconnected vortex particles, other methods use connected vortex line or vortex sheet elements. Figures 5 and 6 show works resulting from vortex line and sheet simulations. In each, a connected triangle mesh is created which defines the surface between two fluids of slightly different densities. This density jump causes creation of vorticity at the surface, which is stored as unique vectorial vortex sheet strengths on each of the triangles. The resulting vorticity acts to contort, fold, and stretch the triangle mesh, which must be constantly adapted to maintain uniform resolution. See Stock [10] for more details about the specific computational method. More similar methods appear in Angelidis & Neyret [3]. Visually similar results appeared later in two-dimensions [2], though the convection was performed using traditional animation methods.

In *Droplet #7* and *Wave For Hokusai* the computational surface itself is rendered in an x-ray-like fashion using a parallel projection, while *Smoke Water Fire* aims to decontextualize the fluid by showing only the sticks defining the raw computational elements. In each of the underlying simulations, the scenarios (initial conditions) were purely artificial, representing no common flow condition. Nevertheless, evolving those systems in time according to the equations presented above generates images with graceful curves and intricate detail and revealing familiar geometric forms. The relatively recent ability of computers and algorithms to replicate shapes and patterns from nature (not limited to fluid flow) gives future artists a new-found ability to communicate by leveraging viewers' own familiarity of those shapes and patterns.

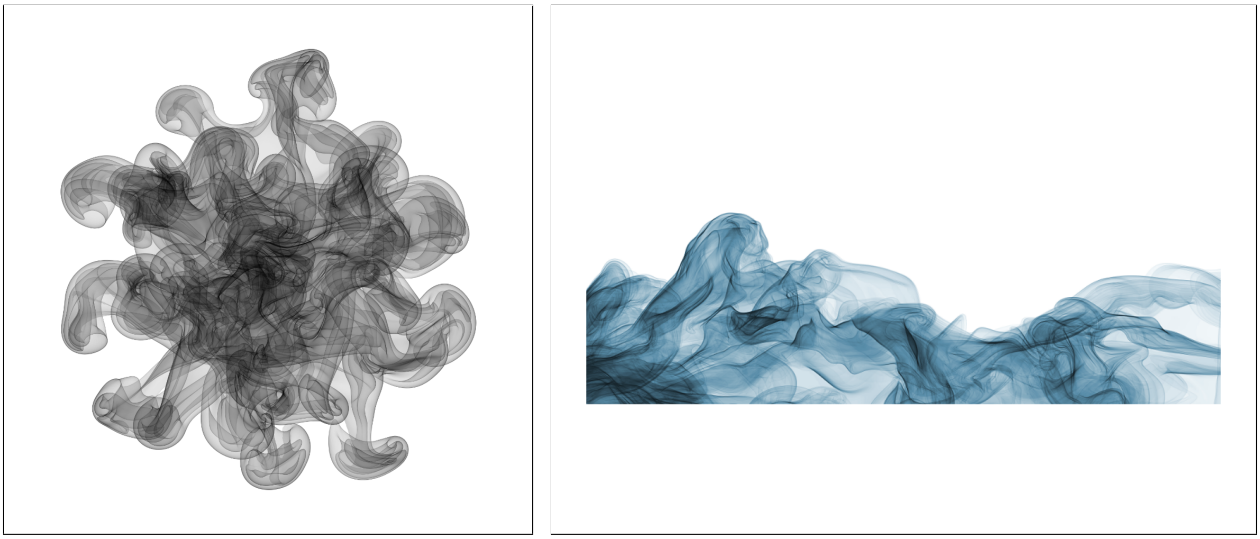

Figure 5 : Droplet #7 *(2004) and* Wave For Hokusai *(2010), examples of 3D vortex sheet simulations.*

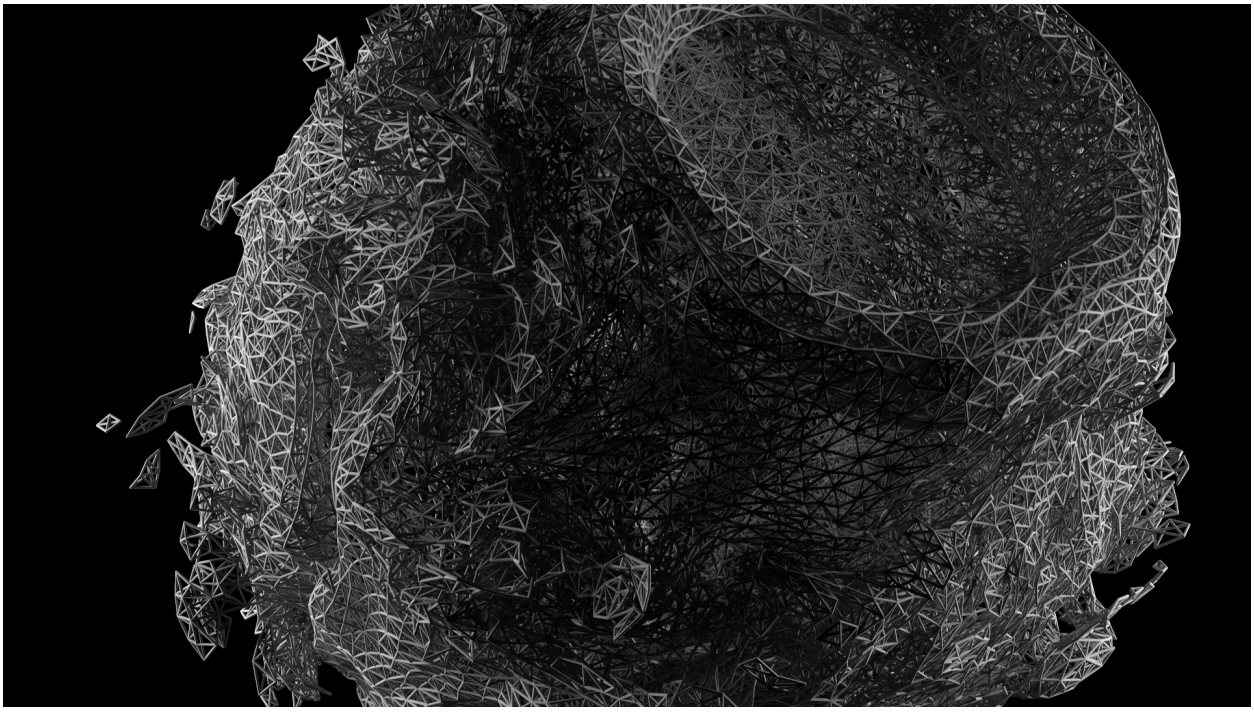

Figure 6 : *Frame from* Smoke Water Fire *(2008), made using a triangulated vortex sheet method.*

4 Conclusion

Improvements in display device technology continue to allow increased information density, and the capacity of human senses to capture and process it all is limited, yet computational resources continue to increase exponentially. The result is that artists are now capable of performing much more work per visual element (pixel or line element) than before. One way to take advantage of that capability is with more complex algorithms. While *ad hoc* exploration of the space of possible algorithms can certainly create novel and rich imagery, we suggest an approach that limits the search to methods that reflect real-world physics. In doing so, it is possible that an artist can more viscerally connect with the viewer by leveraging the viewer's own innate understanding of the physical world. The aim of this paper is not to prove such a conjecture, but to sufficiently introduce the possibilities inherent in computational physics to generate artistic imagery.

Acknowledgements

The author would like to thank Brian Knep for allowing reproduction of his work, and the reviewers for their helpful comments.

References

[1] T.A. Witten Jr. and L.M. Sander. Diffusion-limited aggregation, a kinetic critical phenomenon. *Phys. Rev. Lett.*, 47:1400–1403, 1981.

[2] Ryoichi Ando and Reiji Tsuruno. Vector fluid: A vector graphics depiction of surface flow. In *NPAR '10 Proceedings of the 8th International Symposium on Non-Photorealistic Animation and Rendering*, pages 87–96. ACM, 2010.

[3] Alexis Angelidis and Fabrice Neyret. Simulation of smoke based on vortex filament primitives. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 87–96, New York, NY, USA, 2005. ACM Press.

[4] Robert Bridson. *Fluid Simulation for Computer Graphics*. A K Peters, 2008.

[5] Georges-Henri Cottet and Petros Koumoutsakos. *Vortex Methods: Theory and Practice*. Cambridge Univ. Press, Cambridge, UK, 1999.

[6] M. Gardner. Mathematical Games: The fantastic combinations of John Conway's new solitaire game "life". *Scientific American*, 223:120–123, October 1970.

[7] Charles Hirsch. *Numerical Computation of Internal and External Flows*, volume 1: Fundamentals of Numerical Discretization. John Wiley & Sons, New York, NY, 1988.

[8] A. N. Kolmogorov. The local structure of turbulence in incompressible viscous fluid for very large Reynolds numbers. *Royal Society of London Proceedings Series A*, 434:9–13, July 1991.

[9] Mitchel Resnick. *Turtles, Termites, and Traffic Jams: Explorations in Massively Parallel Microworlds*. MIT Press, 1997.

[10] Mark Joseph Stock. *A Regularized Inviscid Vortex Sheet Method for Three Dimensional Flows With Density Interfaces*. PhD thesis, University of Michigan, 2006.

[11] Stephen Wolfram. *A New Kind of Science*. Wolfram Media, 2002.

[12] Ling Xu and David Mould. Magnetic Curves: Curvature-Controlled Aesthetic Curves Using Magnetic Fields. pages 1–8, Victoria, British Columbia, Canada, 2009. Eurographics Association.