Generative Fluid Profiles for Interactive Media Arts Projects

Angus Graeme Forbes*
School of Information: Science, Technology, and Arts
University of Arizona

Tobias Höllerer†
Dept. of Computer Science
UC Santa Barbara

George Legrady‡
Media Arts and Technology
UC Santa Barbara

Abstract

This paper presents a real-time, interactive fluid simulation and vector visualization technique that can be incorporated in media arts projects. These techniques– referred to collectively as the *Fluid Automata* system– have been adapted for various configurations, including mobile applications, interactive 2D and 3D projections, and multi-touch tables, and have been presented in a number of different environments– both academic and artistic– including galleries, conferences, and a virtual reality research lab. We describe specific details about the fluid simulation component, which, by changing a small number of parameters, allows users to quickly generate a vast number of "fluid profiles" and thus to explore a wide range of aesthetic possibilities that are easy to incorporate into media arts projects. In particular, we present this fluid simulation (and accompanying visual representation) as an example of how media artists can create novel versions of existing visualization techniques in order to emphasize variability, experimentation, and interactivity.

CR Categories: I.3.7 [Computer Graphics]: Methodology and Techniques—Interaction Techniques; J.5 [Arts and Humanities]: Fine Arts—Miscellaneous

Keywords: fluid simulation, vector visualization, video processing, collaborative installation, mobile multimedia art, media art installation

1 Introduction

Scientific visualization projects aim to help researchers identify and reason about salient aspects of their data. While an aesthetic sensibility may contribute to the success of a visualization technique, this is not normally the primary motivation for its creation. Media arts projects, on the other hand, generally place aesthetic considerations at the forefront of their concerns. Similarly, physical simulations are concerned with accuracy and realism rather than extensibility and interaction, which are central to media arts. However, media arts projects, due to time constraints or limitations in technical knowledge, often incorporate readily-available techniques not originally intended for artistic production, and thus that are not necessarily easily adaptable to artistic situations.

This paper presents a series of media arts projects that make use of fluid simulation and vector visualization in a variety of configurations. We describe our implementation of a vector visualization technique and, more thoroughly, the creation of a custom fluid simulation algorithm. We discuss in particular our reasoning when adapting existing techniques in order to make our artworks more useful within a media arts context. Our system has been incorporated in a variety of different projects, including: an iPad application, a virtual reality environment, an interactive collaborative installation, and a multi-touch table. Although the core technology has been featured in variously-named projects, we refer to it as the *Fluid Automata* system hereafter, unless referencing specific aspects of a particular project.

Figure 1: *Photograph of viewers wearing 3D active stereo glasses within an installation of* Annular Genealogy *inside the Allosphere Research Facility at UC Santa Barbara.*

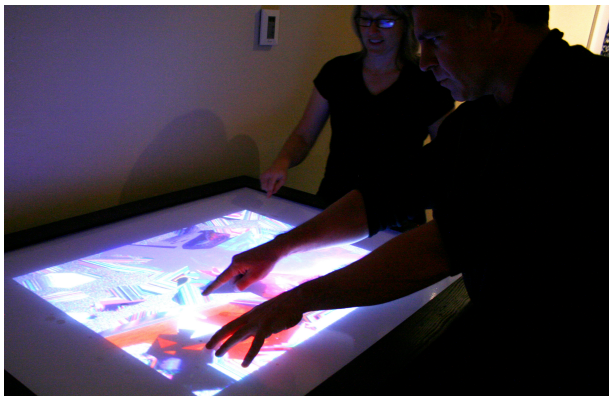

Figure 2: *Users gathered around a multi-touch table running a project that uses the Fluid Automata system.*

The initial implementation of the Fluid Automata system was presented as an interactive generative art system that allowed users to explore the relationship of aesthetics and scientific visualization and the interplay between collaboration and discovery. It was then updated and made available as a downloadable iOS application. This stand-alone mobile application invites users to create

*e-mail:angus.forbes@sista.arizona.edu
†e-mail:holl@cs.ucsb.edu
‡e-mail:legrady@arts.ucsb.edu

CAe 2013, July 19 – 21, 2013, Anaheim, California.

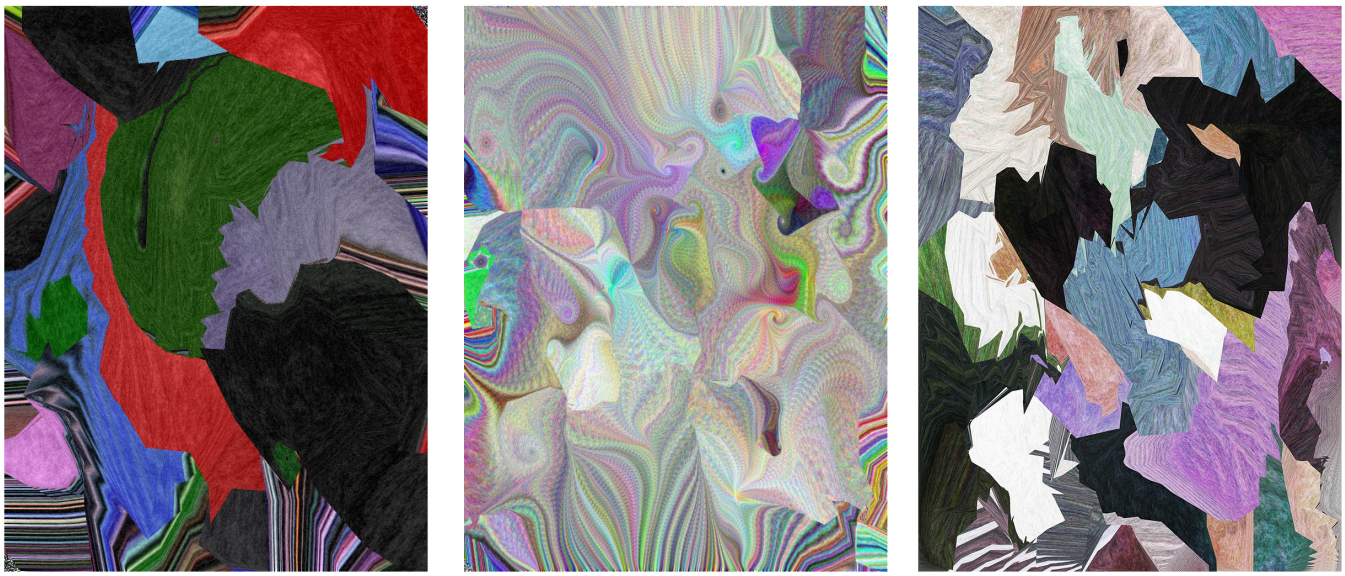

Figure 3: *Examples of output from an iPad application utilizing the Fluid Automata system showing the wide range of aesthetic possibilities generated by the system when using different fluid profiles defined by customizing the fluid, visualization, and noise parameters.*

dynamic generative art via responsive tactile gestures using a tablet computer. The aesthetic experience includes both controlling the system through multi-touch and also adjusting a wide range of parameters to discover new patterns and visual properties; the user can manipulate both the underlying system and its visual representation.

Following those initial implementations, the Fluid Automata system has been presented in a number of different environments, emphasizing different aspects of exploration that are enabled by the project. For instance, one installation emphasizes collaborative experience, inviting multiple users to participate in shaping and interacting with the system, which is projected large-scale onto a wall [Forbes 2011]. The Fluid Automata system has also been incorporated into a visual instrument to provide live accompaniment to a dynamic musical piece created by the composer, Kiyomitsu Odai, called *Studies in Brownian (F*) Motion*. An extension of this project, an audio-visual art piece titled, *Annular Genealogy* (discussed in [Forbes and Odai 2012]), was installed in the AlloSphere Research Facility, a spherical virtual reality environment housed in the California NanoSystems Institute at the University of California, Santa Barbara [Amatriain et al. 2009]. In this project, the tablet's multi-touch, gyroscope, and accelerometer sensors are used to navigate and interact with a fluid system projected on the upper hemisphere of the AlloSphere. Additionally, the tablet can be used to update fluid parameters of the system. Figure 1 shows a still from the *Annular Genealogy* installation. In each of these pieces– whether running on a tablet or a desktop computer– the main components of the Fluid Automata system (the fluid system and the visualization system) run on the GPU using custom GLSL shader programs.

Earlier iterations of this project are described in [Forbes et al. 2012], which provides a general overview of the Fluid Automata system in different contexts, but especially of the the iPad implementation. In this paper, we focus primarily on the fluid dynamics engine, elaborating on the technical details of the system, and indicating how the manipulation of a small set of parameters can generate a wide variety of fluid profiles.

2 Fluid Simulation

A number of interactive art projects use fluid simulation as a component of the work. A method created by Jos Stam in 1999 (and presented to the game developers community in 2003) to create a stable fluid system first made it possible to represent realistic looking fluids at real-time frame rates [Stam 1999; Stam 2003]. Many interactive artworks have made use of this technique. For instance, Memo Atken has created a series of demonstrations based upon Stam's method, showcasing them using mobile devices for interaction and making the code available for OpenFrameworks and Processing multimedia frameworks [Atken 2009]. Another project that incorporates Stam's method is Wakefield and Ji's *Artficial Nature*. This project uses computer vision techniques to allow participants to interact with a 3D fluid representation through the movement of their bodies [Wakefield and Ji 2009]. Other fluid simulation methods, such as [Guay et al. 2011], are optimized for real-time interaction in video games. Although simulation methods generally focus on producing accurate representations of natural systems, the Fluid Automata system demonstrates that aesthetically-interesting visuals with a wide variation of movement and color can be produced from a simple set of rules that do *not* attempt to exactly reproduce natural systems. In this sense, although influenced by physical simulation, Fluid Automata could be considered a "generative art" system [Galanter 2003; Boden and Edmonds 2009].

While the name of the system was inspired (perhaps only poetically) by the biologist Tibor Gánti's discussions of "chemotons" [Gánti 1997; Gánti 2003], the initial kernel of insight for the Fluid Automata system arose while thinking about how to create a simple rule-based system to produce emergent behavior. Cellular automata systems, made popular through the introduction of John Conway's "Game of Life" [Gardner 1970], demonstrate complex behavior emerging through an iterative system that updates the state of each element (positioned in a uniform grid) based on a set of basic rules. These rules determine the next state of each element by querying each of its neighbors. In Conway's original automata system, each pixel has a binary state, and either "lives" (is set to 1) or "dies" (is set to 0) based upon the number of surrounding pixels that are on or off. In many implementations of the Game of Life, users in-

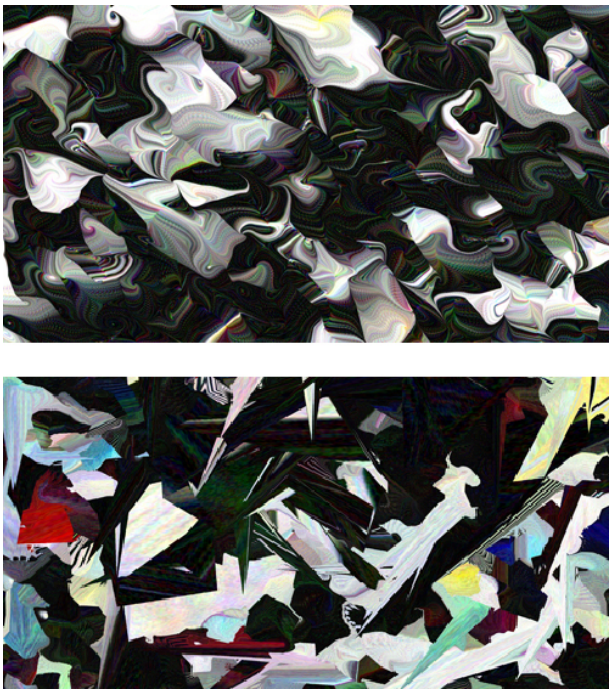

Figure 4: *Details of high-resolution output demonstrating "unrealistic" fluid simulation. The top image shows the addition of a high amount of vorticity; the bottom image shows the "spikiness" associated with a high amount of energy.*

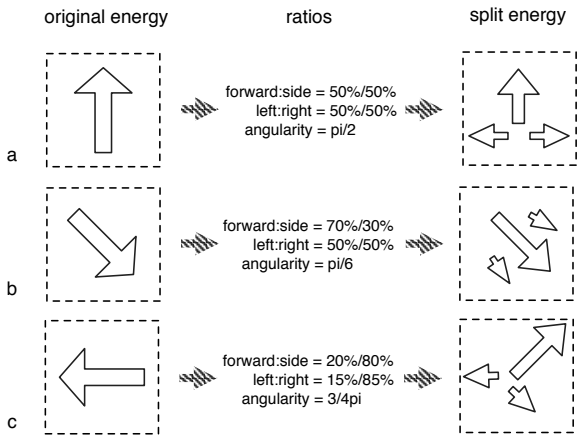

Figure 5: *Examples of fluid systems with different characteristics, i.e., fluid profiles, based on different settings. The settings can be changed in real-time. The left side of the chart represents the energy in a single cell; the right side of the chart shows how the energy is split into different streams based upon the parameters defining the ratio between forward and orthogonal momentum; the ratio between left and right momentum; and the angularity of the orthogonal momentum. In the top row (a), energy with a magnitude of 255 and an orientation of pi/2 is split evenly between forward and orthogonal momentum. In the middle row (b), most of the energy in the cell remains moving in the original direction; the orthogonal energy is close to the forward orientation. In the bottom row (c), most of the energy is moving to the side, with a high angularity, and with an uneven distribution between the left and right sides. (Note, the arrows representing energy vectors are not drawn exactly to scale.)*

teract with the system by using a mouse to turn pixels on or off. Other cellular automata systems, including many that are explored in Stephen Wolfram's "A New Kind of Science" [Wolfram 2002], explore different rule sets and involve multiple states.

Through much experimentation, the current version of the Fluid Automata system effectively simulates the movement of fluid using an 8-bit cellular automata system that stores 256 states for orientation and 256 states for magnitude for each pixel. Retaining the discretization and simplicity central to cellular automata systems allows us to create a complex system that is linear and replicable. Unlike the majority of commonly-implemented fluid simulations which utilize the non-linear Navier-Stokes equations, our algorithm is always stable at any length of timestep. That is, our system is inherently non-realistic due to the fact that there is no mass conservation condition and because the fluid is compressible. Although our system is not physically accurate, it has the advantage of being easy to modify in real-time, and, more importantly, it is easy to comprehend and relatively straightforward to implement, leading to its incorporation in a range of projects.

Since one of the goals in the creation of the Fluid Automata system is to emphasize creativity and interactivity, we created a robust simulation engine that allows a wide range of fluid-like behaviors to be explored. For instance, our system allows users to set parameters describing viscosity, rotational energy, and various momentum parameters. Various versions of this system have been implemented in the different projects that use the Fluid Automata system, taking advantage of available hardware on different devices. But, at its most basic, the system distributes a flow of energy throughout the system as follows:

1. The screen is divided into a grid of cells.
2. New energy is added into the grid by user interaction.
3. The energy at each cell is split into three streams, a forward stream and a left and a right stream.
4. In each of the defined directions for each stream defined in step 3, the energy of each cell is moved into the neighboring cell via the following process:
   (a) The cell is displaced along the vector describing the energy stream.
   (b) For each neighboring cell the displaced cell intersects with, we create a "partial" vector by scaling the original vector with the amount of intersection.
   (c) This partial is added to the cell it intersects with.
5. When this process is finished for the entire grid, the partials associated with each cell are combined, creating a new vector replaces the current vector in the cell.
6. Energy is removed from the system by scaling the energy in each cell by a dampening factor.
7. Steps 2 through 6 are iterated at each timestep until there is no energy left in the system.

More formally, we define a fluid system acting on a grid G of cells $C_{i,j}$ each containing an energy vector $\vec{E}_{i,j}$, with a particular magnitude m and orientation θ, and where $0 < i < columns$ and $0 < j < rows$. The resolution of the grid depends on the effectiveness of the hardware. On a third generation iPad, the maximum resolution at real-time frame rates is a 25x25 grid of cells; on a

desktop computer with a modern graphics card, a 100x100 grid of cells or greater will run at interactive frame rates.

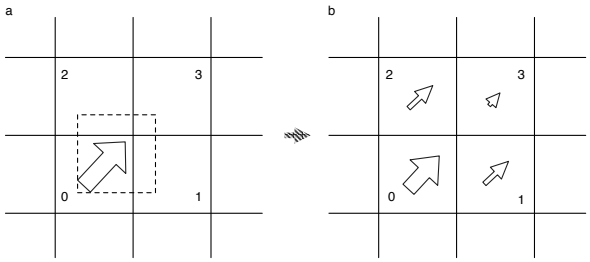

Figure 6: *Example showing how a single stream of energy in a single cell is distributed to other cells. In this example, we see in the left grid (a) that the energy in cell 0 is pointing in the direction pi/4 with a magnitude of 0.5. The dotted box shows where the "displaced" cell intersects with its neighbors. The largest intersection of energy stays within the original cell (cell 0); a tiny amount of energy is pushed into cell 3; and a small amount of energy is pushed into cells 1 and 2. In the right grid (b) we see the distribution of energy from this single cell into its neighbor cells and back into itself.*

In order to define a fluid profile for the system, we define two ratios that regulate that behavior of energy as it moves through the cells. The first, the *momentum* ratio, or r_1, defines how much energy moves forward versus moving to the sides. The second, the *directionality* ratio, or r_2, defines how much energy moves to the left versus moving to the right. We further define a parameter, *angularity*, as $\angle\phi \in [0, \pi)$, describing a rotation offset from $\angle\theta$. Additional parameters influencing the fluid profile are the *viscosity* of the system, which acts as a dampening factor defining the rate at which energy is removed from the system, and the *sensitivity* which controls how much energy is added to the system through some form of user interaction. New energy is added into the cells in a particular direction using the multitouch capabilities of the tablet device. The amount of energy that is added in a particular touch depends upon the *sensitivity* parameter, which can be adjusted during runtime. The magnitude of energy is also determined by how far the current position of the touch is from its previous position. This difference also determines the direction of the added energy. If there is no change in position, then the energy is added in the last known direction. This vector of new energy is added with any existing energy at the currently touched cell to update $\vec{E}_{i,j}$.

At each timestep t during the operation of the fluid system, the energy $\vec{E}$ in $C_{i,j}$ is split into three separate streams: a forward stream, $\vec{F}$, and two "orthogonal" streams, $\vec{L}$ and $\vec{R}$. Using the current fluid profile (the values for the parameters of *momentum, directionality*, and *angularity*), and the current magnitude and orientation for each cell, we define these three streams like so (in Equations 1 through 3):

$$\vec{F} = \begin{pmatrix} r_1 m \\ \theta \end{pmatrix} \quad (1)$$

$$\vec{L} = \begin{pmatrix} (1-r_2)(1-r_1)m \\ \theta + \phi \end{pmatrix} \quad (2)$$

$$\vec{R} = \begin{pmatrix} r_2(1-r_1)m \\ \theta - \phi \end{pmatrix} \quad (3)$$

Figure 5 shows examples of how the current total energy in a cell is split into three streams based on these parameters. Every cell in G thus contains three separate streams of energy, $\vec{F}$, $\vec{L}$, and $\vec{R}$. These streams are used to define the flux of energy moving from each cell into its neighboring cells.

We define a "displaced" cell $D(C, \vec{v})$ as a copy of a cell $C \in G$ that moves along a vector $\vec{v}$ positioned at the center of C. This displaced cell D intersects with between 1 and 4 cells (the original cell C itself and up to three neighboring cells). The magnitude of any energy vector is constrained to range between 0 and 1. When displacing a cell, the magnitude is scaled by the length of a cell side. For instance, if the grid is divided into 10x10 cells, then each cell side is .1 in length. And so the magnitude in a cell is scaled by 1/10th before being displaced. This ensures that a displaced cell will only ever intersect a cell that is its immediate neighbor (although exceptions to this constraint may have interesting creative possibilities.) The amount of overlap between the displaced cell and the cell it intersects with determines how energy is placed into that cell. We define an intersection $I(D, C)$ simply as the amount of overlap between the displaced cell and another cell (Equation 4). The value of any intersection can range between 0 (no intersection) and 1 (full overlap).

$$I(D, C) = \Big(Area(D) \cap Area(C)\Big) / Area(C) \quad (4)$$

And we use this value to create a "partial" vector $\vec{p}$ via a function $P(N, \vec{v}, C)$ for each energy stream. This partial is calculated simply by scaling the stream by the amount it overlaps with the current cell, as defined in Equation 5:

$$P(N, \vec{v}, C) = I(D(N, \vec{v}), C) * \vec{v} \quad (5)$$

In Equation 5, N refers to a neighbor cell of a cell $C \in G$ and $\vec{v}$ refers to one of the energy vectors ($\vec{F}$, $\vec{L}$, or $\vec{R}$) in that neighbor cell. Figure 6 depicts an example of this displacement and the subsequent generation of partials. For each cell $C_{i,j}$ in G, we then examine each neighbor to determine how much each of its three streams overlap. We sum the partials created by any intersection between the neighbor streams to generate the new energy vector for the cell (Equations 6 through 9). It is important to note that we are considering a cell to be a neighbor of itself. For instance, if a vector of magnitude 0.5 is pushed upwards at 90 degrees, it would intersect with both the current cell and the neighbor cell above it. Since the displaced cell would move a distance of 50% of its height from its current position, it would end up intersecting the current cell and the neighbor cell equally, and thus a copy of the vector, scaled by 50%, would be placed in each of the cells. Figure 6 shows a typical example of a single stream of energy in a single cell being distributed to its neighbors. Excluding any dampening factor or any new energy added from user interaction, the system will retain exactly the same amount of energy over each timestep.

$$\vec{F}_{i,j,t} = \sum_{p=i-1}^{i+1} \sum_{q=j-1}^{j+1} P(C_{p,q}, \vec{F}_{p,q,t-1}, C_{i,j}) \quad (6)$$

$$\vec{L}_{i,j,t} = \sum_{p=i-1}^{i+1} \sum_{q=j-1}^{j+1} P(C_{p,q}, \vec{L}_{p,q,t-1}, C_{i,j}) \quad (7)$$

$$\vec{R}_{i,j,t} = \sum_{p=i-1}^{i+1} \sum_{q=j-1}^{j+1} P(C_{p,q}, \vec{R}_{p,q,t-1}, C_{i,j}) \quad (8)$$

$$\vec{E}_{i,j,t} = \vec{F}_{i,j,t} + \vec{L}_{i,j,t} + \vec{R}_{i,j,t} \quad (9)$$

Other parameters can also be adjusted to create different characteristics for a particular fluid profile. These include: controlling the "jitter", or randomness of the system, and clamping the maximum outflow for the cells within the grid. We also experimented with a toroidal representation of the system where fluid energy wraps around the edges of the screen, instead of bouncing off the edges. Setting a maximum outflow parameter interestingly creates the sense of ice cracking and melting when a particular threshold is exceeded. And different settings of *viscosity* and *angularity* can create more or less turbulent behaviors. While it may be surprising that a simple heuristic could mimic the complexity of fluids, the iterative nature of the system does in fact create a wide variety of fluid-like structures, including the creation of eddies, vortices, and turbulence. Figure 3 and Figure 4 show examples of fluid systems with different fluid profiles defined by different settings for the *angularity* and *directionality* parameters.

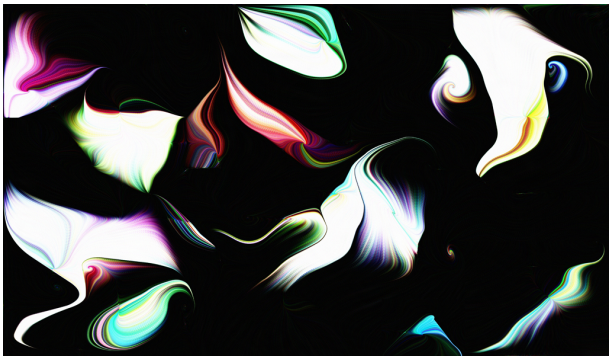

Figure 7: *Example output using a custom color palette with high-contrast image processing variables.*

Just as simulations for realistic films and video games do not feel constrained by a perfect representation of the physics of a visual effect, so to should media artists not feel constrained by a perfect representation of existing algorithms and equations for a particular kind of effect. In our case, by creating our own fluid system with a wide range of parameter adjustments we were able to extend the aesthetic applicability and variation of the fluid system to capture unusual behaviors not normally depicted with fluid representations. Although the system appears realistic, it in fact sacrifices physical accuracy in order to emphasize interactivity, expressivity, and experimentation.

3 Fluid Visualization

A perennial concern of scientific visualization is the effective visualization of salient features of a vector field, as indicated by the wide variety of approaches to their representation [McLoughlin et al. 2010]. A popular technique introduced in 1993, called *Line Integral Convolution*, effectively identifies detailed curvature features of a vector field. In this technique each pixel of a background image is filtered along "streamlines" defined by the vector field [Cabral and Leedom 1993]. Another early technique, *Choreographed Image Flow*, describes using image warping to generate animations for an animated representation of flow-fields [Sims 1992]. A more recent technique, *Image Based Flow Visualization*, represents flow using the iterative deformation of a texture mesh along the directions of the vector fields. In this technique, an image is blended together with the distorted version of itself at each frame [Van Wijk 2002]. While the creators of these techniques recognize and discuss applications outside of scientific visualization, more recent papers more closely examine the relationship between aesthetics and visualization. For instance, [Kirby et al. 2005] specifically looks at the various stylized qualities involved in painting and the possibility of brushstroke techniques for inspiring more effective scientific visualization methods. And [Neyret 2003] introduces a technique that allows artists to control small-scale animations on "advected" textures.

The main image processing scheme in the Fluid Automata system is based on a feedback loop whereby a high-resolution background image is perpetually blended together with a distorted version of itself. The characteristics of the distortion are based directly on the current state of the fluid system. This system is similar to van Wijk's *Image Based Flow Visualization*, which has been extended for use in a variety of scientific visualization applications, including animated and 3D flows [Van Wijk 2002; Telea and van Wijk 2003]. Again, since the focus of the application is aesthetic exploration, we provide the user with a variety of tools to alter aspects of these blending operations, and in addition introduce an image processing layer whereby the user can change a variety of parameters, including: the rate and amount of blending, the type and quality of the background texture, and the brightness, contrast, and saturation of the blended image. The default background texture is a grayscale noise texture at a resolution exactly matching the display size. However, we have experimented with various types of background textures, including lower resolution textures, static colored textures, static image textures, dynamic textures that are updated by noise functions, and dynamic textures that are updated by a live video feed. Figure 7 shows an image created using a static colored background texture with high contrast and high saturation image processing parameters. Figure 11 shows an image with no saturation and that uses a low resolution, black and white background texture to create an interesting "smearing" effect. And Figure 8 shows an image that is created using a live video feed as the background image, rather than an image populated with randomly-colored pixels.

Figure 8: *Photo of iPad application using the Fluid Automata system with a live video feed replacing the background noise texture.*

In Figure 9 we provide an overview of the fluid visualization process over the course of a single frame. At t_i, we: a) distort the previous image texture (if $i > 0$, otherwise we use a copy of the background texture) based on user interaction and the current fluid profile; b) blend in the background texture with the distorted texture based on a blending parameter (that can be updated in real-time by a user); and then c) apply image processing filters (based on the image processing parameters described above) to the image to create a final output texture for this frame at t_{i+1}. This output texture is then used as the input texture for the next frame.

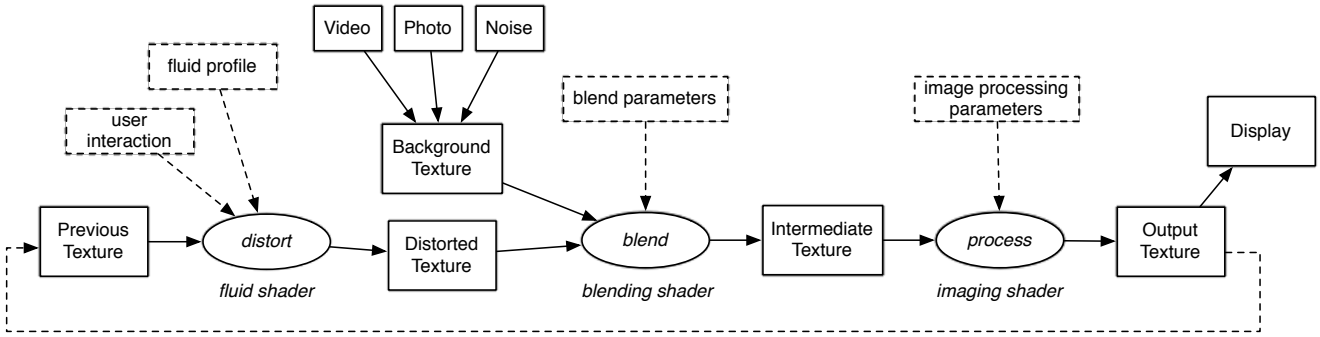

Figure 9: *Schematic for the main functionality of the Fluid Automata system. The output texture after one timestep becomes the input for the next timestep.*

4 Interaction

The Fluid Automata system has been incorporated into a variety of different projects. In earlier projects that used the system, such as a multimedia application for the iPad and an interactive installation involving multiple users, the main interaction modality is the multi-touch interface, used either to interact with a fluid system displayed on the tablet, or to interact collaboratively with a fluid system projected on the wall or within a 3D virtual reality environment [Forbes et al. 2012]. Research investigating interactive flow visualization indicates that novel interaction techniques can successfully enable collaboration and encourage exploration [Isenberg et al. 2009]. Much experimentation went into making the user interaction with the fluid system feel responsive and inviting: by tapping the screen the user adds energy to the system; moving a finger across the screen overrides the fluid dynamic system by forcing the vector to move in the indicated direction; and multiple fingers can be used to push energy around in a more complex way, possibly by more than one person. Other gestures can also be enabled to update fluid properties or image processing parameters. For instance, a pinching gesture using all five fingers simultaneously causes the entire background texture to scale up or scale down, creating a zooming effect. Similarly, a five-fingered panning gesture causes the entire background texture to be translated in the direction of the pan (as determined by the centroid of the five fingers), shifting all of the fluid vectors so that they point in that direction. In addition to being able to add energy to the system, users can update the fluid parameters and image processing parameters in real-time, or select a specified fluid profile defined at a previous time. In a basic version of the project, running on an iPad, users can double-tap the screen to bring up a set of controllers that affect the various parameters of the system. Figures 10a and 10b show a detail of sliders affecting the image processing parameters and fluid parameters, respectively. Users can also save the current fluid profile at any time, allowing it to be quickly retrieved in future sessions. Other types of interaction are specified for the different iterations of the project. For instance, the Fluid Automata system has been ported to a multi-touch table, as shown in Figure 2, to make it easier for multiple people to interact with the system at the same time, and where specific gestures are defined that allow users to alter the intensity of the turbulence of the system.

The Fluid Automata system has also been adapted for use as an instrument for controlling audio-visual compositions. In this configuration the output of the application is projected onto a large display. In addition to being controlled by multitouch, the system can respond to Open Sound Control (OSC) messages sent by another computer that, for instance, are generated by musical events. Additionally, fluid vectors and various fluid parameters can be transmitted wirelessly via OSC to influence the composition. We have also experimented with attaching piezo sensors to the iPad itself in order to directly input data into an algorithmic composition engine. [Forbes and Odai 2012] explores using the Fluid Automata system in combination with musical data.

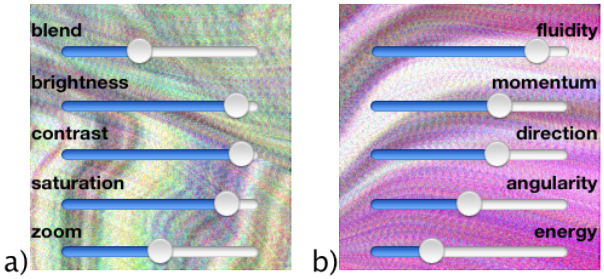

Figure 10: *Details of the iPad controller. The sliders are used to update (a) the image processing parameters and (b) the fluid profile parameters in real-time.*

5 Conclusion

Projects built using the Fluid Automata system exist at the crossroads of visualization and art, using simulation and scientific visualization methods as the basis of interactive, generative art. In presenting our system, we hope that other media artists will be able to use and extend the system for new creative projects. Moreover, we hope that the detailed description of the generative fluid dynamics system demonstrates that there is an appropriate balance between accuracy, realism, and the ease of exploring creative possibilities that can be reached by creators of computational systems. Finally, by describing the steps taken to extend cellular automata into a more extensive system for art production, we hope especially that our system serves to encourage media artists to explore designing their own systems (based perhaps on other simulation algorithms) rather than relying solely on existing techniques that may not be particularly appropriate or adaptable to media arts projects.

Acknowledgements

We thank our colleagues at the University of Arizona, Javier Villegas and Christopher Jette (both in the School of Information: Science, Technology, and Arts), and especially A. M. Proedhel (in the

Figure 11: *Example output of visualization using low-resolution binary background that creates a sharp, smearing effect.*

Department of Computer Science) for providing useful feedback during the writing and editing of this paper.

References

AMATRIAIN, X., KUCHERA-MORIN, J., HOLLERER, T., AND POPE, S. T. 2009. The allosphere: Immersive multimedia for scientific discovery and artistic exploration. *IEEE MultiMedia*, 64–75.

ATKEN, M., 2009. Msa fluid demos. http://www.memo.tv.

BODEN, M. A., AND EDMONDS, E. A. 2009. What is generative art? *Digital Creativity 20*, 1-2, 21–46.

CABRAL, B., AND LEEDOM, L. C. 1993. Imaging vector fields using line integral convolution. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, ACM, 263–270.

FORBES, A. G., AND ODAI, K. 2012. Iterative synaesthetic composing with multimedia signals. In *Proceedings of the International Computer Music Conference (ICMC)*, 573–578.

FORBES, A. G., HÖLLERER, T., AND LEGRADY, G. 2012. Expressive energy: The fluid automata project. In *Proceedings of the International Symposium on Electronic Art (ISEA)*, 65–70.

FORBES, A. G. 2011. *Fluid Automata*. IEEE VisWeek 2011 Art Show Catalog, edited by D. Keefe, B. Campbell, and L. Thorson.

GALANTER, P. 2003. What is generative art? complexity theory as a context for art theory. In *In GA2003–6th Generative Art Conference*.

GÁNTI, T. 1997. Biogenesis itself. *Journal of Theoretical Biology 187*, 4, 583–593.

GÁNTI, T. 2003. *The principles of life*. Oxford University Press.

GARDNER, M. 1970. Mathematical games: The fantastic combinations of john conway's new solitaire game "life". *Scientific American 223*, 4, 120–123.

GUAY, M., COLIN, F., EGLI, R., ET AL. 2011. Simple and fast fluids. *GPU Pro*, 2, 433–444.

ISENBERG, T., HINRICHS, U., AND CARPENDALE, S. 2009. Studying direct-touch interaction for 2d flow visualization. *Collaborative Visualization on Interactive Surfaces-CoVIS'09*, 17.

KIRBY, R. M., KEEFE, D., AND LAIDLAW, D. H. 2005. Painting and visualization. *The Visualization Handbook*, 873–891.

MCLOUGHLIN, T., LARAMEE, R. S., PEIKERT, R., POST, F. H., AND CHEN, M. 2010. Over two decades of integration-based, geometric flow visualization. In *Computer Graphics Forum*, vol. 29, Wiley Online Library, 1807–1829.

NEYRET, F. 2003. Advected textures. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, Eurographics Association, 147–153.

SIMS, K. 1992. Choreographed image flow. *The Journal Of Visualization And Computer Animation 3*, 1, 31–43.

STAM, J. 1999. Stable fluids. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., 121–128.

STAM, J. 2003. Real-time fluid dynamics for games. In *Proceedings of the game developer conference*, vol. 18.

TELEA, A., AND VAN WIJK, J. 2003. 3d ibfv: Hardware-accelerated 3d flow visualization. In *Proceedings of the 14th IEEE Visualization 2003 (VIS'03)*, IEEE Computer Society, 31.

VAN WIJK, J. 2002. Image based flow visualization. *ACM Transactions on Graphics (TOG) 21*, 3, 745–754.

WAKEFIELD, G., AND JI, H. H. 2009. Artificial nature: Immersive world making. In *Applications of Evolutionary Computing*. Springer, 597–602.

WOLFRAM, S. 2002. *A new kind of science*. Wolfram media, Inc., Champaign, IL, USA.