Smoke Brush

Sarah Abraham and Donald Fussell
Computer Science Department at University of Texas at Austin

Figure 1: *Cat drawing with animated smoke effects — generated and constrained by brush stroke.*

Abstract

Non-Photorealistic Rendering covers a wide range of visual effects, and much work has been dedicated to create digital representations of traditional media — either through artist controls or programmatically. We explore a variation of this work, which aims to create digital media that takes the metaphors of traditional media but applies them in ways that have no physical equivalent — thus expanding notions of what digital media can represent. Smoke Brush is a system for applying smoke-like brush strokes to a digital canvas. Using Smoke Brush, artists can add animated, constrained smoke effects to existing pictures or create images represented entirely by smoke. Our drawing system produces artifacts that are realized as animated gifs — a commonly available digital format used in cinemagraphs. We also describe a technique that produces smooth continuous motion in these looped animations that is faithful to the original artist input.

CR Categories: I.3.m [Computer Graphics]: Miscellaneous—Non-Photorealistic Rendering, I.3.4 [Computer Graphics]: Graphics Utilities—Paint Systems;

Keywords: non-photorealistic rendering, smoke, painting, interactive techniques, artistic media

1 Introduction

The area of Non-Photorealistic Rendering (NPR) covers a wide range of techniques and ideas — from digital techniques for artists to programmatic generation of art and stylization. Gooch et al discuss the future of NPR, listing one of its "grand challenges" as designing new artistic media [Gooch et al. 2010], but what constitutes a new artistic medium remains open-ended. One potential area of exploration is understanding how artists can interact with entirely digitally-based media using traditional painting metaphors. Smoke Brush provides a "proof of concept" example as to what a digitally-based medium could constitute, as well as considerations for how artists might interact with such a medium.

Smoke Brush allows an artist to generate smoke-like particles in a localized area of a digital canvas using tablet controls. The stylus provides high-level, intuitive control over a smoke cloud's shape and density, while low-level features, like the individual particle trajectory along the flow is automatic and mostly randomized. The idea of using underlying randomness to create pleasing but unique and recognizable structures has been previously applied for texture generation [Efros and Freeman 2001]. We apply this aesthetic principle to the generation and control of smoke — a fluid used with great effect to enhance visual works with its lively and dynamic movements.

Our system uses the metaphor of paint and brush, where smoke is the "pigment" of the stylus-based brush and an underlying 2D velocity field acts as the paint's "binder" — dispersing the particles in turbulent patterns. By emphasizing simplistic tablet controls with only a few on-screen sliders, we reduce the number of explicit parameters that are often associated with computational fluid dynamics (CFD).

We wish to recreate the real-world experience of painting as the artist generates and controls smoke on a digital canvas, so Smoke Brush runs at interactive rates, which we achieve by emphasizing the aesthetic "feel" of smoke over the physical accuracy of a CFD simulation. In order to provide further visual interest, our systems imports images for the canvas' background. Users can then export seamless animated gifs, which are ideal for cinemagraphs — a popular method of displaying animated art on websites.

2 Previous Work

NPR techniques that recreate traditional media in digital form include everything from charcoal [Bleser et al. 1988] to oils and acrylics [Baxter et al. 2004] to watercolor [Curtis et al. 1997]. What effects digital brushes are able to create outside the realm of traditional media is less explored. Kalnin's WYSIWYG system provides stylistic controls with a more natural painting analogue [Kalnins et al. 2002], but the visual effects are still based on traditional painting. Smoke Brush also aims for an easily understood painting metaphor, but it uses an entirely digital form of "paint." Smoke particles generated in Smoke Brush are restricted to the artist's strokes, and continuously flow in a physically plausible way, while ignoring the physics of heat and particle dissipation —

properties that dictate behavior of smoke in the real world. Since the smoke generated has non-physically-based constraints, Smoke Brush is not equivalent to any traditional art media.

To model smoke effects, it is necessary to understand fluid simulation, which is the standard technique for recreating its distinctive turbulent patterns. Fluid simulation is an area within CFD that models the diffusion and movement of particles. One class of techniques involves an underlying grid of velocity vectors based on the Navier-Stokes equation, which models fluid movements by pushing particles along this flow [Stam 1999]. These particle movements simulate the visually familiar patterns of fluids including water, smoke and fire. Smoke in particular is known for its unique turbulent patterns and vortices [Foster and Metaxas 1997] [Fedkiw et al. 2001].

Fluid simulations also aid modeling of traditional artistic media, such as the diffusion of paint on canvas in Curtis' watercolor technique [Curtis et al. 1997] or the sumi-e system, MoXi [Chu and Tai 2005], which recreate the effects of watercolor and ink respectively and allow users to interact with the medium via a digital canvas and painting interface. Like these systems, Smoke Brush uses an underlying fluid simulation to reproduce smoke-like turbulence in digital paintings, but the effects it generates have no equivalent to a real-world artistic medium, making it distinct.

The question of how to control and constrain fluid has generated its own area of study within the CFD community. Systems can emphasize a high level of control [Treuille et al. 2003][Fattal and Lischinski 2004][Shi and Yu 2005], where the user specifies specific keyframe targets, to mid-level [Barnat et al. 2011], where the user has a general notion of a target, to a low level [Madill and Mould 2013], where the user directs the fluid using parameters based on the underlying physics.

These methods create a variety of unique and visually interesting effects, but our system focuses on "brush-style" control, which attempts to provide a digital analogue to the well-understood paint brush. This is intended to "demystify" the fluid parameters associated with prior control systems and create a highly intuitive, easily mastered program for painters and illustrators of all levels. While Kim et al. created similar controls using paths [Kim et al. 2006], their method does not emphasize the brush metaphor and instead uses 3D NURBS curves. Olsen et al. use interactive vector fields to recreate painterly styles on existing images [Olsen et al. 2005], but this technique is not focused on brush controls or new expressions of artistic media, which is Smoke Brush's primary goal.

Smoke Brush's output is a procedurally-generated series of frames that allows users to create cinemagraphs without looping artifacts. Existing techniques address the problem of seamlessly looping video segments by searching for potential cycles within a sequence [Tompkin et al. 2011] or optimizing each pixel's looping period [Liao et al. 2013]. Video textures use structural analysis of a clip to reorder frames into a seamless loop [Schödl et al. 2000]. In Smoke Brush, the movement of individual smoke particles is inherently random. To match artist intent, the outputted animation only needs to maintain the overall form of the strokes. Thus our technique can avoid the overhead of sequence analysis by overlaying multiple sections of smoke that preserve particle trajectory when the animation loops.

3 Smoke Brush System

Smoke Brush allows an artist to paint smoke particles onto a digital canvas in a way analogous to existing artistic media — namely by using a brush (generation of smoke particles) and canvas (movement of smoke particles). Using a stylus, the artist controls and constrains the general form of the smoke cloud within each stroke. Meanwhile the canvas' underlying velocity field moves particles within the stroke's area in chaotic, turbulent patterns, which make smoke so recognizable as a fluid effect.

The canvas initially is particle-free, and artists can import a background image if they wish to enhance an existing scene with dynamic smoke effects. Artists can create smoke strokes using a stylus as if painting. Similar to how physical paint is comprised of pigment to mark the canvas and binder to diffuse the pigment across the surface of the canvas, the smoke particles have a "pigment" component and a "binder" component. The smoke's pigment is the particles themselves, while the binder is an underlying 2D velocity field, which moves the smoke particles in a turbulent fashion around the canvas. Each particle p has a trajectory defined as its velocity $\overrightarrow{v}_p$, angular velocity $\overrightarrow{\omega}_p$ and transparency α_p. The canvas has an underlying grid of velocity vectors (see Figure 2), and p uses the nearest velocity value to update $\overrightarrow{v}_p$ and $\overrightarrow{\omega}_p$. The system then automatically updates p's position using the semi-implicit Euler method at each time step. Meanwhile α_p modulates based on p's distance from its parent stroke to create a more wispy, visually-pleasing effect.

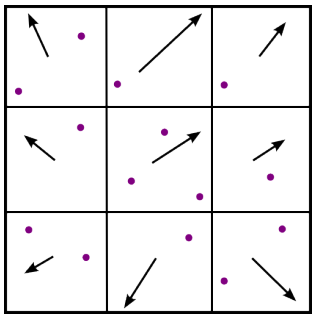


Figure 2: *Underlying velocity grid influences particle position and velocity at every time step.*

Without additional constraints, a particle would move arbitrarily across the canvas. While potentially desirable, this ignores the intent of a brush stroke, which provides directionality and limitations on physically-based paint flow. To achieve these qualities, we discretize strokes into a sequence of "touches," which dictate the behavior of child particles along the stroke. The movements of p are based on the properties of its parent touch, t, which include $strokeWidth_t$, $numParticles_t$ and $lengthScale_t$. $strokeWidth_t$ controls how far child particles can move away from the parent stroke, $numParticles_t$ effects smoke density within t's influence, and $lengthScale_t$ modulates the speed of particles along the underlying velocity field.

3.1 Touch Subdivisions

Particles are constrained to a $strokeWidth$ distance from the parent stroke. While particles should flow freely and smoothly within this area, they remain unaffected by any non-parent strokes, which also helps maintain a sense of stroke cohesion, even when strokes intersect and overlap. Yet simply connecting a stroke's touches into a piecewise series of line segments creates choppy, disjoint sections of smoke, which is undesirable. To create a smoother stroke, we used cubic interpolation to generate subdivisions along the path between touches. The number of subdivisions is dynamically determined during the painting process using the equation:

$$n = \frac{dist(t_0, t_1)}{strokeWidth_{t_0}} \quad (1)$$

where n is the number of subdivisions between touches t_0 and t_1 that ensures an even flow of particles along the stroke regardless of distance between touches. These subdivisions, along with the stroke's touches, form an implicit, continuous path (see Figure 3).

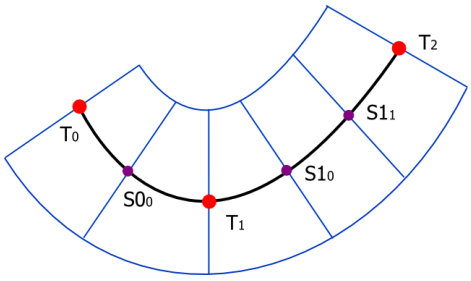


Figure 3: *Subdivisions, spaced $strokeWidth$ distance apart from each other and existing touches, generated along the interpolated stroke.*

3.2 Particle Generation and Placement

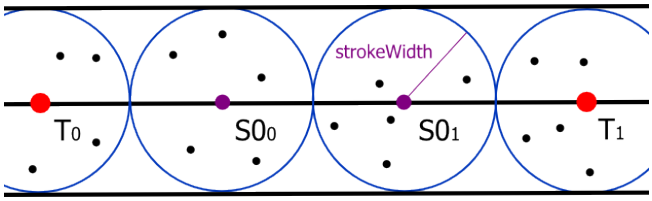


Figure 4: *Particles generated at random within $strokeWidth$ radius around touches or subdivisions. This guarantees particles to be within $strokeWidth$ distance from the stroke.*

Once t's subdivisions are defined, Smoke Brush generates the particles parented by t. For each particle p, we randomly select one of t's subdivisions as a point of origin then randomly generate p's coordinates within a $strokeWidth_t$ radius. This guarantees that p is within the acceptable stroke area (see Figure 4) and simplifies the calculations. While this does not create a mathematically even distribution of particles along the stroke, we found that, due to the nature of the underlying fluid simulation, the particles rapidly disperse into the greater area without deteriorating the final visual output.

To keep p constrained within its parent stroke, S, we must determine whether p is within S's range at each time step. We achieve this by calculating p's perpendicular distance from the nearest subdivision line segment, s, where $s = s_1 - s_0$. To find this distance, we must first calculate b, which is the point along s that is perpendicular to p.

$$b = s_0 + s(\frac{l \cdot s}{s \cdot s}) \tag{2}$$

where $l = p - s_0$. We can then calculate $dist(p, b)$ and compare it to $strokeWidth_t$. Since particles can flow anywhere along the stroke, they are not limited to a particular touch or subdivision. Using $l \cdot s$ and $s \cdot s$, we also determine whether p is beyond the end points of s. We can then reparent p to the nearest neighboring touch as necessary. If p moves beyond the outermost touches of S, we reposition and reattach it to a randomly chosen touch within the stroke, thus preserving particle density while maintaining the touch constraints.

For visual effect we "thin out" the stroke's edges, by linearly fading α_p as it moves away from the stroke curve.

$$\alpha_p = \alpha_p \times (1 - ramp_\alpha) \tag{3}$$

where

$$ramp_\alpha = \frac{dist(p, S)}{strokeWidth_t} \tag{4}$$

and t is p's parent touch within S.

3.3 Curl Noise for Turbulent Effect

Since our goal is a highly responsive, interactive system for artists, we considered methods for fluid simulation that do not necessarily rely on physically accurate calculations. Specifically we used the insights of Bridson's curl-noise technique [Bridson et al. 2007] to generate Smoke Brush's underlying divergence-free 2D velocity field. Curl-noise emphasizes real-time efficiency over physical realism, but like many other fluid simulation techniques, curl noise generates velocity vectors on a grid. These vectors dictate the direction and magnitude of particle trajectories at that given location. At each time-step, $\overrightarrow{v}_p$ and $\overrightarrow{w}_p$ are updated based on p's current velocity and nearest velocity vector along the underlying grid.

Curl-noise uses Perlin noise to model a potential field Ψ. Taking the curl of Ψ creates a turbulent velocity field that remains smooth (no energy sources or sinks that could create numerical instabilities), making it ideal for representing incompressible fluids. This allows for fast, straightforward fluid modeling with easily modulated velocity amplitudes. For our purposes – a 2D medium that runs at interactive rates – we implemented the 2D version. This reduces overall calculations and allows us to move particles across a grid sized at pixel resolutions. On such a fine-grained grid, we're better able to depict the intricate movements of turbulence.

The potential field Ψ extends across the entire canvas, but we want different strokes to have different properties within it, as well as variation within the stroke itself. To accomplish this, each parent touch t independently determines how Ψ effects its child particles. t's length scale ($lengthScale_t$) modifies Ψ's magnitude for $\overrightarrow{v}_p$, while $strokeWidth_t$ determines p's maximum distance from the parent stroke, S. This allows the artist to generate a variety of stroke effects using the same underlying Ψ.

We considered constraining p by modulating Ψ with a smoothing ramp (also discussed in the Bridson paper), but this results in particles slowing as they move away from S, whereas we expect an even speed across all particles as the smoke emerges and dissipates. Thus we perform a simple check of $dist(p, S)$ at each time step and reposition p if necessary.

3.4 Noise over Time

Using a single Perlin noise map eventually results in particles "stabilizing" within cyclic eddies, which is not a desirable effect. To prevent this, we randomly generate Perlin noise every few seconds and linearly interpolate Ψ from the existing noise field to the new noise field. Since Ψ remains smooth during this interpolation, the velocity field also remains incompressible and stable. Particles can then move smoothly along the velocity field with no jumps or shifts between Ψ modulations for indefinite periods of time. This also ensures each picture has its own, unique patterns to the underlying flow.

4 Additional Controls

Smoke Brush aims to create an intuitive, easy-to-learn interface for artists with limited experience in fluid simulation. We found

the tablet/stylus interface best suited our needs, and our GUI provides sliders only when fine-grained control outweighs the need for a brush-based interface. The only two slider controls are Stroke Width, which determines an initial value for $strokeWidth$, and Smoke Speed, which modulates $lengthScale$. A checkbox, Restrict Smoke, informs the system whether a touch's particles are constrained by $strokeWidth$, or whether they should flow freely across the entire canvas.

The stylus implicitly provides further control over the smoke. Stylus pressure determines the number of particles (i.e. particle density) and also modulates $strokeWidth$ for a more natural line effect. This is based on traditional painting models, where an artist would use brush pressure to modulate both width and pigment quantity, creating visually pleasing variation along a stroke.

The stylus' eraser removes both particles and touches within the eraser stroke's radius. For simplicity, we remove all particles associated with an erased touch as well, but it's possible to reparent these to neighboring touches if desired.

5 Exporting Seamless Animation Loops

Since Smoke Brush aims to create animated smoke effects, artists can output their work as a series of frames, which can be reconstructed as either a cinemagraph or short movie. Ideally this sequence should loop seamlessly to create a smooth, high-quality animation, but creating a seamless loop is non-trivial. Since particles are repositioned throughout the simulation, there is not necessarily any frame-to-frame coherency in terms of particle position.

In the interests of computational efficiency and simplicity, we use the following insights: the particles already have random trajectories, so the output will not change qualitatively if we overlay additional particles with similar trajectories, and the appearance of density will also not change qualitatively if we modulate these overlaid particles' transparencies. Thus we can use these qualitative equivalences by overlaying particles from different time steps to smoothly transition between the animation's last and first frame.

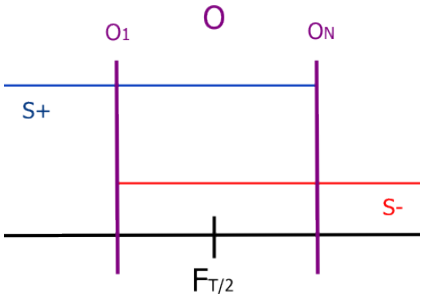
First, we capture particle information from a sequence of frames F, starting at F_1 and ending at F_T, where T is the total number of frames in the desired animation loop. We additionally capture particle information from frame sequences of length S before and after F. $S-$ is the sequence F_{1-S} to F_1, and $S+$ is the sequence F_T to F_{T+S}. Since looping F creates a discontinuity between frames F_T and F_1 as the particles return to their initial positions, we take the particle information from $S-$ and $S+$ and overlay them on F, aligning $S+$ starting at F_1 and $S-$ starting at F_{T-S} (see Figure 5). This eliminates the discontinuity between F_T and F_1 as the particles in F_T continue on their current trajectories in F_1 and the earlier trajectories of particles in F_1 are visible in frames leading up to F_T.

Figure 5: *Frames over time showing the particle trajectories within frames F_{1-S} to F_1 and F_T to F_{T+S} shifted to their respective positions within the captured sequence of frames — F_1 to F_T. This provides a smooth transition in the particle trajectories from F_T to F_1 when the sequence loops.*

While not quantitatively equivalent to the initial output, this technique maintains the overall consistency of individual strokes. The density of particles roughly doubles, but the appearance of density is reduced by lowering each particle's alpha across F, $S+$ and $S-$.

Overlaying $S+$ and $S-$ onto F does lead to discontinuities where $S-$ begins and $S+$ ends (similar to our initial problem transitioning between F_T and F_1). But we correct this by choosing S such that $S > \frac{T}{2}$. Now $S+$ and $S-$ overlap across the time period represented by frames O as shown in Figure 6. This makes the discontinuities less noticeable, as now only a third of the particles have a discontinuity at any given time step. Yet the density is again no longer consistent, so a visible increase in particles occurs within O. We correct for this visual artifact by cross-fading $S-$ and $S+$, so that $S-$ fades in at the same rate $S+$ fades out. This reduces the appearance of a density increase within O and leads to a smooth transition across all frames.

Figure 6: *Particles from $S-$ and $S+$ cross-fade within the overlap region O. By calculating a step size based on $max(p_\alpha)$ within $S+$ at frame O_1 and the $max(p_\alpha)$ within $S-$ at frame O_N, we linearly interpolate p_α in $S-$ to 0 at frame O_1, and p_α in $S+$ to 0 at O_N.*



5.1 Cross-fade

For this cross-fade to appear continuous, we must select an appropriate alpha step size to apply to particles within $S-$ and $S+$. Within each frame, particles have unique α values, so for a true linear interpolation, not only does each particle need a unique alpha step size, but we also need to track the particle between frames for temporal coherency. To avoid this overhead, we instead find a single step size value for α_{S-} and α_{S+} within the "most opaque" frame of S in O. For $S-$, this is the frame at O_N where N is number of frames in O, while for $S+$, this is its frame at O_1 or the first frame in O (see Figure 6).

α_{S-} and α_{S+} are then the maximum α_p represented within this frame. We then use these to calculate the alpha step size.

$$\alpha_{S_{step}} = \frac{\alpha_S}{N} \tag{5}$$

for both α_{S+} and α_{S-} independently.

We can now linearly interpolate along the frames in $S+$ and $S-$, adjusting α_p according to:

$$\alpha_p = \alpha_p - (\alpha_{step} \times t) \tag{6}$$

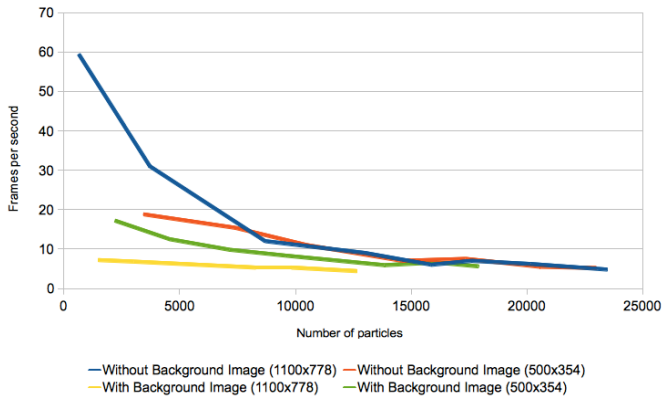
where t is frame position in O. We experimented with cosine interpolation, but linear interpolation provided smoother transitions. Using S of length 15 frames or more provided a pleasing, imperceptible transition.

6 Results

We tested a single-threaded version of Smoke Brush on a 2.16 GHz Intel Core Duo MacBook Pro with 2 GB of RAM. The system's performance varies whether or not the user has loaded in an existing image to paint on, suggesting optimizations within QT (the GUI

framework used) could benefit Smoke Brush's overall performance. For our initial experiments, we used a small library of hand-drawn textures to render the smoke particles, and the system ran at around 20 frames per second with 3000 particles simulated (roughly two or three brush strokes), but this frame rate drops as particles are added. In most of test cases, the system could handle over 17000 particles before dropping below 5 frames per second (see Figure 7).

Figure 7: *Frames per second based on total number of particles across canvases with and without a pre-loaded background image.*



6.1 User Study

As an initial user study, we asked 5 artists (some with digital background and some with traditional) to use our system. Responses were positive, and they agreed the novelty of the experience was enjoyable. Controls were intuitive, and the familiarity of the stylus interface created a workflow similar to what even traditional artists used in their regular work. Within a few test sketches, they were able to grasp controls well enough to produce satisfying artifacts.

Figure 8: *A photograph of a train enhanced with animated smoke painted in a single brush stroke ©Chelsea Hostetter.*

Users expressed interest in the fluid nature of the medium, while also finding it a challenge — particle movements creating a "looser" feel than usually expected with brush placement. The uniqueness of this interaction also invited different approaches to strokes and content. Especially with the "free-drawn" smoke pictures, artists favored organic, flowing content like the tree in Figure 10 or the specter in Figure 11.

Figure 9: *A photograph of an octopus enhanced with an animated ink jet created with few brush strokes ©Chelsea Hostetter.*

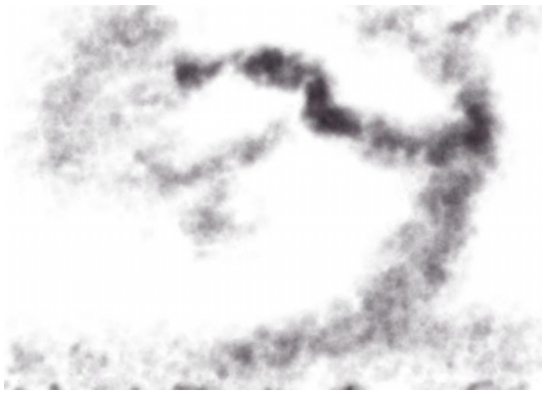


Figure 10: *A free-drawn tree made entirely in Smoke Brush using numerous brush strokes ©Katie Bauer.*

It is possible to insert smoke in places where smoke is a natural fit, such as the steam train example in Figure 8, but it can also be used as a substitute for other turbulent fluids, like the octopus' ink jet in Figure 9. Artists also added smoke to digitally drawn images like the phoenix in Figure 12 or the smoke stacks in Figure 13.

Suggestions for future iterations of Smoke Brush included more precise controls as an option and the standard digital controls, such as masking capabilities to enforce strict "no smoke" boundaries, color palette control and layers.

7 Future Work

Smoke Brush is a very initial exploration into a form of new artistic media and intuitive system for drawing fluids. We intend to incorporate feedback from the initial user study to increase the range of effects possible, including customizable smoke color and textures for smoother, more natural results. A higher quality selection of professionally drawn textures could potentially lead to more visually pleasing results, but rendering has not been the focus of our work thus far and will require more thorough analysis in future iterations.

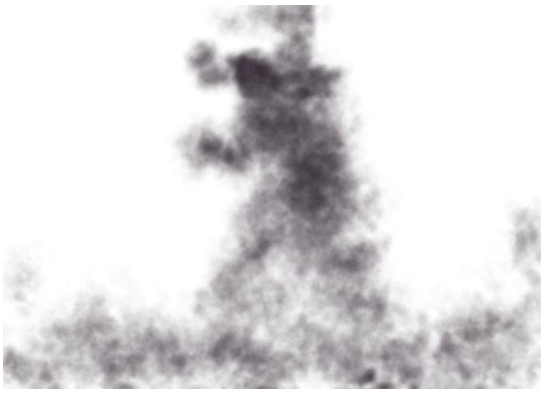


Figure 11: *A free-drawn abstract specter made entirely in Smoke Brush ©Chelsea Hostetter.*

Figure 12: *A digitally drawn phoenix enhanced with smoke clouds ©Lauren Liebowitz.*

Another important aspect to examine is the user controls and interface. Thus far we have only explored the stylus' pressure controls, which maps to the real-world analogue of brush pressure, but stylus tilt and stroke velocity have real-world analogues within painting as well. We intend to explore these avenues to create greater functionality and increase the intuitiveness of the controls.

More generally, Smoke Brush focuses on rendering smoke-like effects, but the core idea — and implementation — allows for a general-purpose system that takes velocity data from any underlying fluid simulation. Therefore we intend to experiment with other types of fluid, such as water, fire, and clouds, and potentially explore other motions as well. We believe capturing directional movement of particles in addition to the localized turbulence will expand the range of effects possible within our system. Such extensions will likely require new techniques to seamlessly loop the animated output as well.

Figure 13: *A digitally drawn factory enhanced with smoke ©Evan Kight.*

Acknowledgements

Artists/Photographers: Chelsea Hostetter, Katie Bauer, Lauren Liebowitz, Jeff Isacksen, Evan Kight.

References

BARNAT, A., LI, Z., MCCANN, J., AND POLLARD, N. S. 2011. Mid-level smoke control for 2d animation. In *Proceedings of Graphics Interface 2011*, Canadian Human-Computer Communications Society, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, GI '11, 25–32.

BAXTER, W., WENDT, J., AND LIN, M. C. 2004. Impasto: A realistic, interactive model for paint. In *Proceedings of the 3rd International Symposium on Non-photorealistic Animation and Rendering*, ACM, New York, NY, USA, NPAR '04, 45–148.

BLESER, T. W., SIBERT, J. L., AND MCGEE, J. P. 1988. Charcoal sketching: Returning control to the artist. *ACM Trans. Graph. 7*, 1 (Jan.), 76–81.

BRIDSON, R., HOURIHAM, J., AND NORDENSTAM, M. 2007. Curl-noise for procedural fluid flow. In *ACM SIGGRAPH 2007 Papers*, ACM, New York, NY, USA, SIGGRAPH '07.

CHU, N. S.-H., AND TAI, C.-L. 2005. Moxi: Real-time ink dispersion in absorbent paper. In *ACM SIGGRAPH 2005 Sketches*, ACM, New York, NY, USA, SIGGRAPH '05.

CURTIS, C. J., ANDERSON, S. E., SEIMS, J. E., FLEISCHER, K. W., AND SALESIN, D. H. 1997. Computer-generated watercolor. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, SIGGRAPH '97, 421–430.

EFROS, A. A., AND FREEMAN, W. T. 2001. Image quilting for texture synthesis and transfer. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, ACM, New York, NY, USA, SIGGRAPH '01, 341–346.

FATTAL, R., AND LISCHINSKI, D. 2004. Target-driven smoke animation. *ACM Trans. Graph. 23*, 3 (Aug.), 441–448.

FEDKIW, R., STAM, J., AND JENSEN, H. W. 2001. Visual simulation of smoke. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, ACM, New York, NY, USA, SIGGRAPH '01, 15–22.

FOSTER, N., AND METAXAS, D. 1997. Modeling the motion of a hot, turbulent gas. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, SIGGRAPH '97, 181–188.

GOOCH, A. A., LONG, J., JI, L., ESTEY, A., AND GOOCH, B. S. 2010. Viewing progress in non-photorealistic rendering through heinlein's lens. In *Proceedings of the 8th International Symposium on Non-Photorealistic Animation and Rendering*, ACM, New York, NY, USA, NPAR '10, 165–171.

KALNINS, R. D., MARKOSIAN, L., MEIER, B. J., KOWALSKI, M. A., LEE, J. C., DAVIDSON, P. L., WEBB, M., HUGHES, J. F., AND FINKELSTEIN, A. 2002. Wysiwyg npr: Drawing strokes directly on 3d models. *ACM Trans. Graph. 21*, 3 (July), 755–762.

KIM, Y., MACHIRAJU, R., AND THOMPSON, D. 2006. Path-based control of smoke simulations. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, SCA '06, 33–42.

LIAO, Z., JOSHI, N., AND HOPPE, H. 2013. Automated video looping with progressive dynamism. *ACM Trans. Graph. 32*, 4 (July), 77:1–77:10.

MADILL, J., AND MOULD, D. 2013. Target particle control of smoke simulation. In *Proceedings of the 2013 Graphics Interface Conference*, Canadian Information Processing Society, Toronto, Ont., Canada, Canada, GI '13, 125–132.

OLSEN, S. C., MAXWELL, B. A., AND GOOCH, B. 2005. Interactive vector fields for painterly rendering. In *Proceedings of Graphics Interface 2005*, Canadian Human-Computer Communications Society, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, GI '05, 241–247.

SCHÖDL, A., SZELISKI, R., SALESIN, D. H., AND ESSA, I. 2000. Video textures. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, SIGGRAPH '00, 489–498.

SHI, L., AND YU, Y. 2005. Controllable smoke animation with guiding objects. *ACM Trans. Graph. 24*, 1 (Jan.), 140–164.

STAM, J. 1999. Stable fluids. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, SIGGRAPH '99, 121–128.

TOMPKIN, J., PECE, F., SUBR, K., AND KAUTZ, J. 2011. Towards moment imagery: Automatic cinemagraphs. In *Proceedings of the 2011 Conference for Visual Media Production*, IEEE Computer Society, Washington, DC, USA, CVMP '11, 87–93.

TREUILLE, A., MCNAMARA, A., POPOVIĆ, Z., AND STAM, J. 2003. Keyframe control of smoke simulations. *ACM Trans. Graph. 22*, 3 (July), 716–723.