Pacific Graphics 2013
B. Levy, X. Tong, and K. Yin
(Guest Editors)

Volume 32 (2013), Number 7

Interactive Physics-based Ink Splattering Art Creation

Su-Ian Eugene Lei[1] Ying-Chieh Chen[1] Hsiang-Ting Chen[2] Chun-Fa Chang[3]

[1]National Tsing-Hua University
[2]JST ERATO Igarashi Design Interface Project / The University of Tokyo
[3]National Taiwan Normal University

Abstract

This paper presents an interactive system for ink splattering, a form of abstract art that artists splat ink onto the canvas. The default input device of our system is a pressure-sensitive 2D stylus, the most common sketching tool for digital artists, and we propose two interaction mode: ***ink-flicking mode*** *and* ***ink-dripping mode****, that are designed to be analogous to the artistic techniques of ink splattering in real world. The core of our ink splattering system is a novel three-stage ink splattering framework that simulates the physics-based interaction of ink with different mediums including brush heads, air and paper. We have implemented the physical engine in CUDA and the whole simulation process runs at interactive speed.*

Categories and Subject Descriptors (according to ACM CCS): I.3.4 [Computer Graphics]: Graphics Utilities—Paint systems

1. Introduction

Ink splatter is an important element in artwork creation process. The vibrant pattern of the splattering ink not only convey vigor, dynamism and grittiness but also serve as a source of inspiration for art work creation. Artists usually create the splattering pattern by forcefully flicking the brush toward the paper or hovering the paint-dripping brush over the paper. While highly experienced artists may have the skill to harness the subtle shape and motion of splatter, the ink splatter's nature of being highly irregular and unpredictable makes the creation of the desired pattern a time-consuming trial-and-error process.

For such an iterative creation task, a digital painting tool with undo and redo functions is usually desired by the artists. However, the closest function that modern image editing software provides so far is the digital brush with pre-defined sets of splatter pattern, which usually exhibit visible repetition. Skilled artists can hide the repetition by constantly switching between different patterns or carefully fine-tune the jittering parameter associated with each brush stroke, yet it is a time-consuming process and the inspiring randomness of ink splattering is mostly lost.

Pioneering research prototypes tackle the problem with simplified physical simulation. Lee et al. [LOG06] focus on the specific painting style of Jackson Pollock and Chu et al. [CT05] ignore ink dynamics in the air but focus on the interaction between ink and paper. However, our experiments show that the creation of realistic and organic splattering patterns requires a physical simulation that fully considers the physical aspects of the ink splattering process (Figure 2), which consists of the factors of viscosity and surface tension

Figure 1: A sample picture created with our system.

Figure 2: The appearance of even a simple brush stroke is shaped by fluid flow and surface tension. Top: A brush stroke created by flicking the virtual brush in our system. Bottom: A similarly shaped stroke created in real-life.

of ink, ink distribution on the brush, air dynamics of ink, and ink-paper interaction.

We present an interactive physics-based ink splattering system consisting of a novel three-stage ink splattering simulation framework and a dual-mode user interface imitating real-world ink splattering skills.

The three-stage simulation framework simulates the following real-world ink splattering process:

- **ink-brush stage** that simulates the brush dynamics and ink movement within brush.
- **ink-air stage** that tracks the movement of ink within the air.
- **ink-paper stage** that simulates the complex interaction between ink flow and paper.

In particular, we utilize the graphics processors (GPUs) for the simulation of Lagrangian fluid [GSSP10, HBD10] to simulate a wide variety of art materials, from highly viscous oil paint to thinly diluted watercolor. To further enhance the realism, we also simulate the flowing of pigments and fluid on different paper materials that occurs after the ink hits the paper.

The two interaction modes are:

- **ink-flicking mode** that simulates the brush motion when artists quickly flick ink onto the paper, when stronger dynamic and randomness are desired.
- **ink-dripping mode** that simulates the dripping or splashing of ink onto the paper with an ink-filled container, where precision is the priority.

Our main contribution is a physics-based ink splattering system to facilitate the creation of ink splatter art. Our contribution goes beyond the integration of fluid simulation and the dual interaction modes. By giving the artists a better control of the desired splattering patterns and a quick feedback on the results, they are empowered to explore expressions that are tedious or error-prone in real world or previous digital tools.

2. Previous Work

Paint and Paper Simulation: Physically-based simulation of real world art materials that goes beyond simple image processing technique is a commonly approached topic in non-photorealistic rendering [CAS*97, CT05, BWL04, EWK*13, DKMI13, WW07, JCM07]. Baxter et al. [BWL04] provide a realistic simulation model for oil and acrylics paint. However the mixture of paint mostly occurs when the artist manually pushes different colored paints around with the brush. The Moxi system of Chu et al. [CT05] simulates Eastern ink drawings using a physics based fluid model to carry and mix pigments. Their work is able to create realistic ink effects such as flow patterns without manual guidance from the artists.

While those approaches realistically simulate the paint behavior, they mostly simulate the physical characteristics after the paint is applied onto the paper and canvas. Recent works such as [CT04, BG10] model the interaction between brush, paint and paper extensively. However in order to simulate paint flicking, the effect of brush and paint movement by air resistance and brush inertia must also be modeled.

Ink Splattering Simulation

Chu et al. [CT05] provide a splash and spray tool for Moxi, which uses the brush as a particle emitter throwing blobs at the paper. The movement of the blobs follows basic Newtonian physics, leaving elliptic footprints on the paper and depositing the ink with no fluid simulation in the air. Also the blobs are directly "shot" from the brush, since brush deformation in the air is not simulated. While this tool is simple, this still creates a good effect due to the solid ink and paper simulation, with the flow of pigments between ink drops deposited in different timesteps creating intricate patterns.

Lee et al. [LOG06] focus their simulation on recreating the drip painting artworks of Jackson Pollock. They use a one-dimensional fluid jet to simulate paint movement in the air. The reason behind this is that Jackson Pollock often uses paint with very high viscosity for his artworks, and the one-dimensional fluid jet is a good approximation. However this only applies when the paint is delivered in a gentle dripping motion. It does not handle the case when the paint is violently thrown onto the canvas to create dramatic splattering patterns.

In contrast, our system simulates the physical interaction of paint with brushes, air and paper in real time. Users can interactively adjust the parameters of paint material such as viscosity and surface tension.

Fluid Simulation: The two most widely used approaches for computational fluid mechanics are Eulerian (grid based) and Lagrangian (particle based). While both methods simulate a wide range of materials [HK05, CMT04, CBP05], a grid-based approach would require a very dense 3D grid [HK05] to simulate paint dynamics in the air. A practical choice for interactively modeling highly deformable fluid is by smoothed particle hydrodynamics (SPH). SPH models fluid with a set of particles of which their physical properties are smoothed by kernel functions. Until recently

SPH systems can only simulate a small amount of particles (less than 1000) in real time. Recent development in programmable graphics processor (GPU) makes it possible to operate on large amount of particles in an interactive frame rate [GSSP10, HBD10]. Our application uses an SPH solver implemented in CUDA [Buc07] to model the ink spilled from the brush to the point where it hits the paper.

The Lattice Boltzmann method (LBM) [Suc01, WZF*04] is an increasingly popular alternative approach for grid-based fluid simulation. LBM models the flow of a Newtonian fluid by simulating the streaming and collision of frictive particles. An advantage of LBM is that all operations are simple and local. Each update of cell requires only information from its immediate neighbors (similar to a celluar automata), making this very efficient to implement on parallel graphics processors. Chu et al. [CT05] used this method to model paint flow on the paper. We base our paint and paper interaction on their work since it best simulates the looks of watercolor and Chinese ink splatters, and we extend their paper model to include surface friction, glazing and absorption when an ink drop hits the paper.

Brush Simulation: Saito et al. [SN99] model brush dynamics in 3D by approximating the bristle behavior using energy optimization on a single spine. Baxter et al. [BG10] and Chu et al. [CT04] extend this method to simulate the characteristics of different brush types such as the Chinese calligraphy brush.

These methods mainly concern the appearance of the brush footprints on the canvas. Therefore their main focus is the interaction between brush and paper, and generally neglects brush deformation in the air caused by external forces such as gravity and air resistance. While this is acceptable in some cases where external forces are negligible comparing to brush elasticity, the quick and forceful motion of paint flicking will cause significant deformation of brush hair. And the movement of brush hair indeed affects the resulting splattering pattern. We implement a brush model based on a spring-mass system that simulates the full 3D deformation of brush hair in the air according to the user movement of the brush.

User Interface for Digital Sketching: The majority of digital artist uses graphics tablet for artwork creation. These tablets support variable pressure and tilt input. The interface of [BG10] and [CT04] uses these input to deform the paintbrush (such as spreading the brush tip hair outwards) when the brush head touches the canvas.

Graphics tablet does not provide 3D location tracking of the stylus, so special interface design (or hardware) is required if the artist need to manipulate paint material in 3D. Our physical brush interface uses the pressure of the stylus to simulate the z-axis, letting the user either hover the brush above the canvas or create quick smacking motions based on the pressure applied.

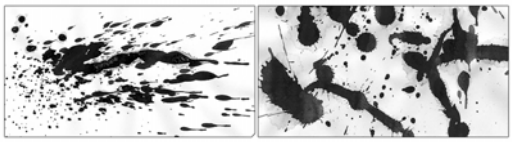

Figure 3: Two common splattering patterns. Left: Outward scattering patterns. Right: Continuous trails.

Lee et al. [LOG06] employ a simpler interface by letting an artist create a paint bucket object in 3D space, then guide it using a 2D tablet. We have implemented a similar ink-jet interface, by setting the height of the brush, then controlling the paint being splashed out based on tablet input.

3. Interaction Mode

In real world, artists create the splattering art with two types of motion: flicking and dripping. For the former, the artist flicks the brush at high speed toward the paper, which results in an outward scattering pattern with different distribution of droplets depending on the flicking angle. For the later, the artist loads the brush (or any type of container, e.g. a spoon) with a large amount of paint, then let it naturally drips onto the canvas. While gravity does most of the work, one can direct the flow in mid-air by subtle motions. Figure 3 shows splatters created by these two types of motion.

To model these motions using the graphics tablet and its stylus, first we need to understand the input data they can provide. A graphics tablet and its stylus provide five degrees of freedom: the stylus location (in 2D), tilt, bearing (both in degrees) and pressure on the tablet. The stylus controls a cursor in 3D space at location $\mathbf{x}$, where x_x and x_y corresponds to the 2D coordinates of the stylus on the tablet. The velocity of the stylus, $\mathbf{q}_{velocity}$, is calculated by measuring the displacement of $\mathbf{x}$ between two timesteps. At each timestep, the coordinates $\mathbf{x}$, $\mathbf{q}_{velocity}$, $\mathbf{q}_{tilt}$ and the pressure $q_{pressure}$ are sampled from the stylus. Using these data, we can devise a 3D controlling scheme for both the ink-flicking and the ink-dripping interfaces.

3.1. Ink-Flicking Mode

The first interaction mode emulates a virtual brush, which correlates directly to the stylus motion. Besides the flicking motion, the amount of ink loaded by the brush also affects the resulting pattern. We design a paint bucket interface, where users can dip the brush and control the loaded ink amount. In order to simulate 3D motions with a pen tablet, we use the stylus pressure to control the height of the brush (Figure 4).

$$x_z = c_{height}(1.0 - q_{pressure}) \quad (1)$$

where

$$0 \leq q_{pressure} \leq 1 \tag{2}$$

and c_{height} is a variable that denotes the maximum height of the brush. The orientation and angle of the brush directly reflects the tilt of the stylus $\mathbf{q}_{tilt}$.

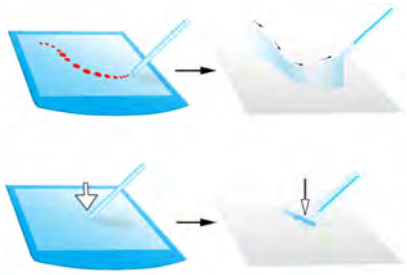

Figure 4: Up: The input we received from the graphics tablet, and its corresponding path of the brush. The size of the red dots denotes the pressure applied on the tablet by the stylus. Down: A sudden application of pressure creates a fast smacking motion.

3.2. Ink-Dripping Mode

The ink-dripping mode emulates a stylus-controlled ink-jet that dispenses ink based on the stylus angle. Similar to the common digital painting software, the stylus pressure controls the size of the paint nozzle, with $x_z = c_{height}$ which is controlled using the mouse wheel. The amount of paint being released is directly proportional to $q_{pressure}$. To create "directional" splats, we use $\mathbf{q}_{tilt}$ to control the orientation of the ink-jet. We also use $\mathbf{q}_{velocity}$ to affect the velocity of the jet to simulate momentum (Figure 5).

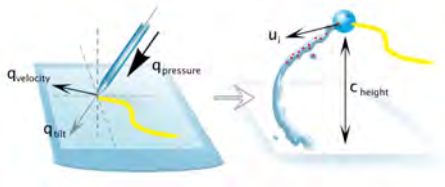

Figure 5: Left: The input we received from the graphics tablet. Right: The corresponding path of the ink-jet, with $\mathbf{u}$ being the velocity of emitted particles and c_{height} a user defined parameter.

Sometimes it is more desirable to let the artist control where the paint would land on the paper, while the location of the paint source would be calculated accordingly. In this mode, the stylus controls a cursor on the paper, which indicates the location where the paint would ultimately land. As in the basic configuration, the pressure and tilt of the stylus control the size and direction of the paint jet. This in turn gives the control over the final size and general splash direction of the splatter pattern. Given the target position $\mathbf{y}$, the position of the paint jet $\mathbf{x}$ is calculated by:

$$x_z = c_{height} \tag{3}$$

$$x_{(x,y)} = y_{(x,y)} + u_{(x,y)} t_{drop} \tag{4}$$

$$t_{drop} = \frac{2x_z}{u_z + \sqrt{u_z^2 + 2gx_z}} \tag{5}$$

t_{drop} indicates the estimated time it takes for a particle to fall on the paper after being emitted from the paint jet. We follow the basic equations of motion for its simplicity. While individual particles created with this method may not land exactly at the target, the drop path of the fluid volume does not deviate too much from basic equations of motion. Therefore this is an acceptable way to create splatter patterns at pre-defined positions. Figure 6 shows ink splatters following roughly the same paths but created with different parameters.

Figure 6: Three splatters following roughly the same dripping path. Left: With the stylus tilted left, creating splatters mainly to the left side. Center: With the stylus tilted right. Right: With highly viscous paint. Notice the lack of splatters.

4. Simulation Framework

Our simulation framework consists of three consecutive stages:

- Ink-brush stage: We initialize the simulation based on the information gathered from the user interface. If the ink-flicking interface is chosen, this stage also simulates the dynamics of brush head and loaded paint.
- Ink-air stage: We simulate the trajectory of splattering inks in the air and their self-interaction using the smoothed particle hydrodynamics (SPH) model, necessary in creating realistic splattering patterns.
- Ink-paper stage: When the ink hits the canvas, we simulate the dispersion and absorption between the ink and the paper using the Lattice-Boltzmann Method (LBM) model.

Figure 7 shows an overview of the process.

4.1. Ink-Brush Stage

Before the fluid particles are created, we need to simulate the physical behavior of the brush. We use a spring-mass system to simulate our brush head. Brush deformation is modeled

Figure 7: The process flow of our system.

using individual spines, where each spine is a series of connected springs. In a spring-mass system, each node n_i has a mass of m_{n_i} and position $\mathbf{x}_{n_i}$. Each spring segment i (between nodes n_i and n_{i+1}) has a rest length of r_i (Figure 8).

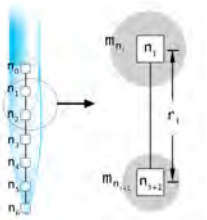

Figure 8: Left: A spine of the brush head. Right: The spring-mass system of a single segment, where m is the mass of a node, and r is the rest length of the segment.

Basically, each node is susceptible to an external force $\mathbf{F}$ such as gravity. Therefore at each timestep, the displacement of node i is calculated by:

$$\Delta\mathbf{x}_{n_i} = (1.0-\zeta)\Delta_{last}\mathbf{x}_{n_i} + \mathbf{a}_{n_i}t^2 \tag{6}$$

$$\mathbf{a}_{n_i} = \frac{\mathbf{F}}{m_{n_i}} \tag{7}$$

where $\Delta_{last}\mathbf{x}_{n_i}$ is the displacement of $\mathbf{x}_{n_i}$ in the last timestep, and t is the length of the timestep. $\zeta < 1.0$ is a damping term, which heuristically accounts for the gradual loss of velocity due to air resistance and friction.

If the current length of the segment $l_i = |\overline{\mathbf{x}_{n_i}\mathbf{x}_{n_{i+1}}}|$ does not equal to the rest length r_i, a correction vector $\mathbf{d}_i$ is applied to move both node back into place:

$$\mathbf{d}_i = (\mathbf{x}_{n_i} - \mathbf{x}_{n_{i+1}})(1 - \frac{r_i}{l_i}) \tag{8}$$

$$\mathbf{x}^*_{n_i} = \mathbf{x}_{n_i} + \mathbf{d}_i \tag{9}$$

$$\mathbf{x}^*_{n_{i+1}} = \mathbf{x}_{n_{i+1}} - \mathbf{d}_i \tag{10}$$

Equations 8,9,10 are looped through all nodes for $c_{stiffness}$ times to incrementally correct the position of the nodes so the distance between consecutive nodes converges to rest length r_i. The variable $c_{stiffness}$ denotes the stiffness of the spring-mass system. The higher the value, the less elastic the brush hair is.

To model the load of the paint brush, each node i contains a volume of absorbed paint V_i, with a maximum of $Vmax_i$. Also we assign the weight of the brush hair m_{hair_i} to each node. Therefore with a paint of density ρ_0, we have:

$$m_{paint_i} = \rho_0 V_i \tag{11}$$

$$m_{n_i} = m_{hair_i} + m_{paint_i} \tag{12}$$

In our brush model, the paint is dislodged from the brush head once the force of a node $\mathbf{F}_i = m_{n_i}\mathbf{a}_i$ exceeds a threshold ξ_i. The amount of paint leaving brush node i would be:

$$m_{dislodge_i} = m_{paint_i} - \frac{\xi_i}{\mathbf{a}_i} \tag{13}$$

Figure 9 shows the process of paint dislodging. The threshold ξ_i is a function of brush hair sorptivity $S(\frac{m}{\sqrt{s}})$, paint viscosity $\mu(\frac{kg}{m\cdot s})$, surface tension $\sigma(\frac{N}{m})$ and the current segment length l_i(m). Sorptivity is related to the density and inner-friction of the bristles of the brush, while viscosity is the internal friction of the fluid. Both values inversely affect the ease of paint dislodging. Surface tension also pulls the liquid inwards to prevent dislodging. Therefore the threshold can be written as:

$$\xi_i = c_{d_1}S^2\mu + c_{d_2}\sigma l_i \tag{14}$$

where c_{d_1} and c_{d_2} are dislodging constants that affect the amount of paint leaving the brush based on viscosity, tension and sorptivity.

Also, the paint within the brush hair slowly seeps downward due to gravity. For each timestep:

$$\Delta V_i = V_i \cos\theta_i \cdot c_{seep} \tag{15}$$

$$V^*_i = V_i - \Delta V_i \tag{16}$$

$$V^*_{i+1} = V_{i+1} + \Delta V_{i+1} \tag{17}$$

where c_{seep} is the rate of downward seeping, and θ_i is the angle between segment i and the z-axis. If $V_k > Vmax_k$, where k is the last node (the tip of the brush), the paint that exceeds the limit is also dripped down from the brush.

Now that we know the amount of paint that dislodges from the brush at each timestep, we need to create its corresponding particles for the SPH simulation. In SPH, the initial placement of the particles is crucial to the stability of the sys-

tem. The particles must be uniformly distributed according to the rest density (ρ_0) and particle mass (m) of the fluid:

$$d = \sqrt[3]{\frac{3m}{4\pi\rho_0}} \tag{18}$$

We use a simple approach to initialize the particles. At each node that $m_{dislodge_i} > 0$, we create a cubic blob of uniformly distributed particles with length $l_{c_i} = \sqrt[3]{\frac{m_{dislodge_i}}{\rho 0}}$ at each side. To avoid overlapping of particles (thus introducing instability), a particle is only generated if there are no other particles within its interaction radius h. The initial velocity of the particles within a cube is equal to the velocity of the node that dislodges the paint. The blocky appearance of the initial fluid volume is quickly smoothened by the SPH simulation in about 3 timesteps (Figure 9).

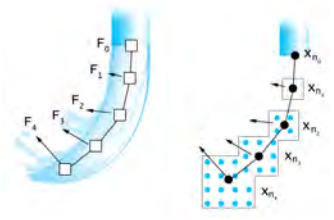

Figure 9: Left: The movement force of each node in the brush head, and an approximate representation of ink dislodge amount. Right: The actual particles created based on this amount.

4.1.1. Ink-Dripping

The ink-brush stage for the ink-dripping interface is much simpler than the ink-flicking interface. The tilt of the stylus controls the direction of the paint leaving the brush, while the pressure controls its size and power. The velocity of the stylus movement also affects the direction and power of the paint thrown, to simulate momentum. Therefore, a set of particles are initialized with the initial velocity $\mathbf{u}$:

$$\mathbf{u} = (c_{pressure} q_{pressure} + c_{tilt}) \times \mathbf{q}_{tilt} + c_{vel}\mathbf{q}_{vel}\frac{|\overline{\mathbf{x}_i\mathbf{x}_{i-1}}|}{t} \tag{19}$$

Where $\mathbf{x_i}$ is the position of the paint jet at timestep i, $c_{pressure}$, c_{tilt} and c_{vel} are user adjustable parameters which control how much tilt and stylus velocity affect the jet direction and power, $\mathbf{q}_{tilt}$, $\mathbf{q}_{vel}$ and $q_{pressure}$ are the tilt vector, velocity vector and pressure of the stylus respectively, and t is the length of the timestep (Figure 5).

We provide an option to let the user create a continuous stream of ink or a string of droplets while creating a brushstroke in the air. For a continuous stream we use the method described by [Str86] to generate brushstrokes in 3D space. We then fill each segment of the stroke uniformly with particles using a basic scanline method (Figure 10). To create a stable initial condition, the fluid must have a 3D volume. We achieve this by simply replicating the original set of particles along the z-axis, with a depth of $h_i = \frac{cP_i + cP_{i+1}}{2}$, as shown in Figure 10, where c is a scalar controlling the brush width, and P_i and $P_(i+1)$ are the stylus pressure of two consecutive samples.

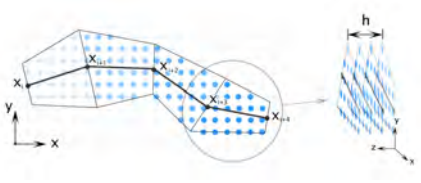

Figure 10: A sample brushstroke. Each slice corresponds to a sample from the graphics tablet.

4.2. Ink-Air Stage

In the ink-air stage, we simulate the ink dynamics in the air and its splattering on the paper. The paint is defined by a set of particles initialized in Section 4.1, where each particle i carries mass(m_i), position($\mathbf{r}_i$) and velocity($\mathbf{u}_i$). The first step in SPH simulation is to calculate the density at each particle site i.

$$\rho_i = \sum_j m_j W(\mathbf{r}_i - \mathbf{r}_j, h) \tag{20}$$

where W is the smoothing kernel and h is the interaction radius. Then the pressure at i is determined.

$$f_i^{pressure} = -\sum_j \frac{m_j}{\rho_j}\frac{p_i + p_j}{2} \bigtriangledown W(\mathbf{r}_i - \mathbf{r}_j, h) \tag{21}$$

$\bigtriangledown W$ is the gradient of the smoothing kernel. p_i is the pressure at particle i, where $p_i = k\rho_i$ and k is the internal stiffness of the liquid. Viscosity of the fluid, which is caused by internal friction between particles [MCG03], is calculated by

$$f_i^{viscosity} = \mu\sum_j (\mathbf{u}_j - \mathbf{u}_i)\frac{m_j}{\rho_j} \bigtriangledown^2 W(\mathbf{r}_i - \mathbf{r}_j, h) \tag{22}$$

where $\bigtriangledown^2 W$ is the Hessian of the kernel function, and μ is the viscosity coefficient, which defines the strength of how viscous the fluid is. By modifying this value we can simulate different types of paint ranging from oil paint to thin watercolor. The definition of W, $\bigtriangledown W$ and $\bigtriangledown^2 W$ can be found in [Kel06].

In the last stage of the simulation, the velocity of each particle is updated.

$$\frac{d\mathbf{u}_i}{dt} = \frac{\mathbf{F}_i}{\rho_i} \tag{23}$$

where $\mathbf{F}_i$ is the sum of internal forces (pressure, viscosity) and external forces (gravity, surface tension).

Each particle, depending on the choice of material, also represents a mixture of an amount of water (ρ_a) and pigment (p_a). These particles run through the SPH simulation in Equations 20 to 23. When a particle reaches the canvas surface, there is a chance that it would be absorbed by the canvas. If it is not, it would bounce back when its velocity is sufficiently high:

$$\mathbf{u}_i^* = \mathbf{u}_i - (1+c_R)(\mathbf{u}\cdot\mathbf{n})\mathbf{n} \tag{24}$$

where $0 < c_R < 1$ is the coefficient of restitution that dampens the bounce back velocity. If the velocity along the z-axis (the vector pointing directly away from the canvas) is below a certain threshold, the particle will be "stuck" to the canvas, its velocity dampened by friction of the current site for each iteration. Each particle below a certain height on the z-axis is considered "touching" the canvas, and leaves behind a footprint according to Section 4.2.1 until it is completely absorbed by the canvas.

We model the absorption of the particle by using its age. The absorption rate of the current site λ on the paper generally represents the amount of paint absorbed in each timestep. We simply add this rate divided by current frame-per-second to the particle's age until it is larger than 1.0, at which point the particle will be considered completely absorbed and be removed from the simulation.

4.2.1. Point Splatting

The shape left behind by each particle on the canvas determines how realistic the end result would be. Using point splatting [Wes91] to render the particle footprints with the following strategy produces good result:

$$\mathbf{u}_{plane}(i) = \begin{bmatrix} u_x(i) \\ u_y(i) \end{bmatrix} \tag{25}$$

$$\mathbf{S}_1(i) = \frac{\mathbf{u}_{plane}(i)}{|\mathbf{u}_{plane}(i)|} \tag{26}$$

$$\mathbf{S}_2(i) = \mathbf{S}_1(i)^{\perp} \tag{27}$$

$$r_{\mathbf{S}_2}(i) = 1.0 + u_z(i) \times c_{dropvel} \times rand(0,1) \times c_{pressure}\sqrt{f_i^{pressure}} \tag{28}$$

$$r_{\mathbf{S}_1}(i) = (1 + c_{planevel} \times |\mathbf{u}_{plane}(i)|) r_{\mathbf{S}_2(i)} \tag{29}$$

For each particle i, $\mathbf{u}_{plane}(i)$ is the velocity projected on the x-y plane (the canvas). The splat is an ellipse with its major axis $\mathbf{S}_1(i)$ aligned to $\mathbf{u}_{plane}(i)$. The lengths of its major and minor radii are $r_{\mathbf{S}_1}(i)$ and $r_{\mathbf{S}_2}(i)$ respectively (Figure 11).

The reason behind this strategy is that in that each particle holds a certain amount of fluid. We use $u_z(i)$ to determine whether this particle impacts on a paper or simply grazes it. In the case of an impact, the fluid volume represented by the particle is squashed and flattened on the paper, leaving a bigger footprint; whereas in the latter case, the particle leaves a smaller footprint since surface tension pulls the volume into a spherical shape. In either case, the splat is also stretched along $\mathbf{u}_{plane}(i)$.

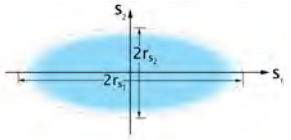

Figure 11: A splat generated by a particle.

However, when a cluster of particles hits the canvas, many particles will end up hitting and be absorbed at the same position, creating many overlapping splats. To get a more realistic result, we need to determine the size of the cluster since a larger one naturally will leave a bigger footprint. Particle pressure $f_i^{pressure}$ is a good indicator since a higher value means the particle is deeper within the fluid, therefore implying that it belongs to a larger cluster.

Using this strategy, we can create highly realistic splatters that accurately reflect real-world behavior, such as the bowling-pin and crown shaped drops (Figure 12). The user can control how much $u_z(i)$, $f_i^{pressure}$ and $|\mathbf{u}_{plane}(i)|$ affect the splatting size using the parameters $c_{dropvel}$, $c_{pressure}$ and $c_{planevel}$ respectively. A comparison between our method, uniform-sized splats and no splatting is shown in Figure 13.

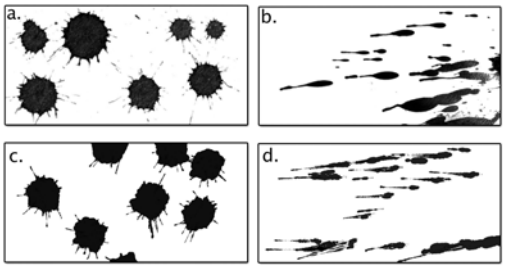

Figure 12: Comparison between splatter pattern created in real-world (a, b) and in our system (c, d). (a) Crown shaped drops created by dripping paint on the paper. (b) Bowling-pin shaped drops created by flicking paint at an angle. (c)(d) Similar patterns created using our system.

4.3. Ink-Paper Stage

When the ink hits the paper, we use the LBM fluid model to simulate its interaction with the canvas. A fluid simulation at this stage is necessary to simulate the more complex flow patterns, since simple diffusion cannot account for

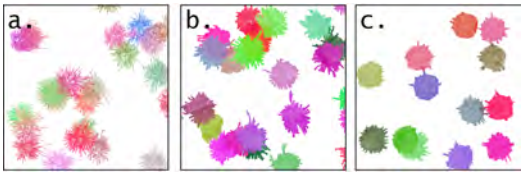

Figure 13: Comparison between circular drops created with different splatting approaches. (a) Without splatting. (b) Uniform-sized splatting. (c) Our splatting strategy based on both particle speed and pressure.

cases when pigment flow is impeded by paper fibers or accumulation of ink constituents [CT05].

In LBM models, the fluid is composed by pseudo particles, which perform propagation and collision process over a regular lattice mesh. We use a 2D uniform grid for our simulation, which is classified as the D2Q9 model. Each cell contains nine distributions f_i, which denote the expected numbers of particles moving along lattice vectors e_i (Figure 14). For each timestep, two operations are performed at each cell:

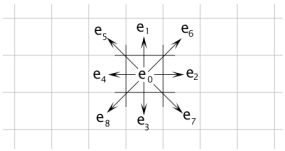

Figure 14: The D2Q9 lattice model and the 9 vectors of a sample cell.

Propagation: f_i are propagated to the neighboring cells along the vectors e_i.

Collision: f_i that arrive at the same site are collided and redistributed toward their equilibrium functions $f_i^{(eq)}$.

The propagation and collision in the D2Q9 are described as:

$$f_i(\mathbf{x}+\mathbf{e}_i\triangle t, t+\triangle t) = (1-\omega)f_i(\mathbf{x},t)+\omega f_i^{eq}(\mathbf{x},t) \quad (30)$$

In the incompressible fluid modei, $f_i^{(eq)}$ is

$$f_i^{(eq)} = w_i\left\{\rho+\rho_0\left[\frac{3}{c^2}\mathbf{e}_i\cdot\mathbf{u}+\frac{9}{2c^4}(\mathbf{e}_i\cdot\mathbf{u})^2-\frac{3}{2c^2}\mathbf{u}\cdot\mathbf{u}\right]\right\} \quad (31)$$

where w_i is the weight. w_i is $\frac{4}{9}$ for $i=0$, $\frac{1}{9}$ for $i=1,2,3,4$ and $\frac{1}{36}$ for $i=5,6,7,8$. c is the propagation speed on the lattice, which we set to $c=1$. Fluid density ρ and velocity $\mathbf{u}$ are simply determined by

$$\rho = \sum_{i=0}^{8} f_i \quad (32)$$

$$\mathbf{u} = \frac{1}{\rho}\sum_{i=1}^{8}\mathbf{e}_i f_i \quad (33)$$

Our paper model is seperated into two main layers: the footprint layer and the simulation layer. The footprint layer has a much higher resolution since ink splattering creates highly detailed patterns. The simulation layer closely follows those defined by [CT05], which further separates the paper into three sub-layers: *surface*, *flow* and *fixture*.

When the particle touches the paper, it creates a footprint using point splatting on the footprint layer. This splat is then down-sampled and converted into a composition of water and pigment in the surface layer. This composition gradually seeps into the flow layer. The flow layer is where the fluid simulation takes place. Water flows (and evaporates) within this layer according to the LBM process (Equations 30 to 32). Pigment redistributes to adjacent cells based on f_i (particle distribution along vector $\mathbf{e}_i$):

$$\mathbf{p}_\mathbf{f}^*(\mathbf{x}) = \frac{1}{\rho}\sum_{i=1}^{8} f_i \mathbf{p}_\mathbf{f}(\mathbf{x}-\mathbf{e}_\mathbf{i}) \quad (34)$$

where $\mathbf{p}_\mathbf{f}$ and $\mathbf{p}_\mathbf{f}^*$ is the amount of pigment in the flow layer in the current and next timestep respectively. For each iteration a certain amount of pigment of the flow layer is transferred to the fixture layer, where they no longer move along with the fluid.

5. Implementation

All our results on this paper were generated using a desktop computer with an Intel Core i7 CPU, an NVIDIA GeForce GTX 680 GPU, and a Wacom Intuos 2 graphics tablet. Our SPH code is based on the open sourced FLUIDS v.3 by R.C. Hoetzlein [Hoe12], released under the Z-lib license. We also used CUDA to implement the LBM fluid simulator, and used OpenGL and NVIDIA Cg 3.1 for the final rendering. In our examples, we used a canvas with a resolution of 4096^2, and the LBM paper-and-ink simulation were performed on a resolution of 512^2. Each brush stroke is simulated with 8000 to 16000 particles in SPH. We have an overall system frame rate of 40 to 70 frames per second, depending on the number of particles currently being simulated in SPH.

6. Discussion

We conducted a pilot study on our system with several professional artists. Figures 1 and 17b are the artworks created by them, where each creation process took less than 15 minutes. In Figures 1 and 17b, the artists used the ink-dripping interaction to apply splatters around a stencil mask (a popular splattering creation skill in real world) and create highly stylized artworks. We summarized their feedback in the following section.

6.1. User Interface and Resulting Patterns

Our collaborating artists found the ink-dripping mode easier to adopt to. Because in this mode, the cursor, which indicates where the inks would land, moved along the paper; the control is similar to the ordinary digital paintbrush function. However those who have experience in ink splattering art prefer the more intense and dynamic patterns generated with ink-flicking mode. For example, in Figure 15a and 15b we show two splatter patterns created by thrusting the stylus forward under the ink-flicking and ink-dripping mode. The ink-dripping mode generates a more stable and regular pattern whereas the ink-flicking mode generate the patterns with more subtle variation in orientation, speed and distribution of the droplets.

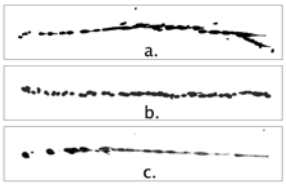

Figure 15: Comparison between splatters created with different modes (a) ink-flicking mode (b) ink-dripping mode (c) splatters in real-world.

The reason is that the action of ink splashing involves vigorous brush movement, which in turn causes vibration in brush hair, splashing paint in a way that is hard to replicate when using the ink-dripping interface. Figure 15c shows a real-world splatter example created by flicking a loaded brush. The resulting pattern shows characteristics similar to those created with the ink-flicking interface.

6.2. Comparison with Other Methods

Comparing with Chu et al.'s splash and spray tool [CT05], our 3D fluid simulation creates much more realistic and complex splatters. Combined with an ink-paper model that takes into account the grazing and absorption of individual ink drops, we can create inter-flowing splattering patterns as shown in Figure 17b.

In Lee et al.'s work [LOG06], they used a statistical model to create drops and splatters in user defined locations. This model creates a number of "fingers" based on the velocity and diameter of the drop, creating crown or claw shaped patterns. Our full 3D simulation can create much more realistic and organic splattering patterns (Figure 16).

Also, as discussed in the previous section, an ink-dripping interface similar to the approach of [LOG06] may not be adequate to convey the full range of dynamics shown in Pollock's artwork. Our ink-flicking interface, by simulating a full 3D range of brush motion, can create the subtle effects like those in Figure 15 which are often observed in ink splattering artworks.

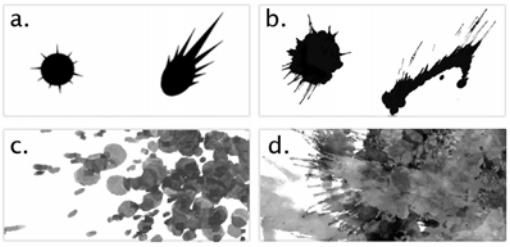

Figure 16: Comparison with other methods. (a) Sample splatters created using Lee's statistical model. (b) Sample splatters created using our full 3D fluid simulation and splatting. (c) Splatters created using Chu's splash and spray tool. (d) Splatters created using our system.

7. Conclusion

We present an interactive system for ink splattering art. The *ink-dripping* and *ink-flicking* interaction modes imitate the real-world artistic skills; artists can extend their real-world exprience and adapt to our digital system easily. The core three-stage ink splattering framework simulates the interaction of ink-brush, ink-air and ink-paper. We show that our system could generate more realistic organic patterns by comparing with previous works and real-world splatters. In the future, we will explore the possibility of using the 1D fluid simulation of [LOG06] to more accurately model paint movement within the brush hair, in order to create a better initial particle distribution when paint dislodges from the brush.

8. Acknowledgements

This work is supported in part under the "Embedded software and living service platform and technology development project" of the Institute for Information Industry which is subsidized by the Ministry of Economy Affairs (Taiwan), and by National Science Council (Taiwan) under grant NSC 102-2219-E-003-001.

References

[BG10] BAXTER W., GOVINDARAJU N.: Simple data-driven modeling of brushes. In *Proc. Symposium on Interactive 3D Graphics (I3D 2010)* (2010). 2, 3

[Buc07] BUCK I.: Gpu computing with nvidia cuda. In *ACM SIGGRAPH 2007 courses* (2007), SIGGRAPH '07. 3

[BWL04] BAXTER W. V., WENDT J., LIN M. C.: IMPaSTo: A realistic model for paint. In *Proc. of Symposium on Non-Photorealistic Animation and Rendering (NPAR)* (June 2004), pp. 45–56. 2

[CAS*97] CURTIS C. J., ANDERSON S. E., SEIMS J. E., FLEISCHER K. W., SALESIN D. H.: Computer-generated watercolor. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques* (1997), SIGGRAPH '97, pp. 421–430. 2

(a) Bunny, drawn using our system.

(b) Ink splattering art using two masks. We apply precise splatters in user defined locations using the backward projection technique. This showcases the use of ink splatters as a design element.

Figure 17: Sample pictures created with our system

[CBP05] CLAVET S., BEAUDOIN P., POULIN P.: Particle-based viscoelastic fluid simulation. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2005), SCA '05, pp. 219–228. 2

[CMT04] CARLSON M., MUCHA P. J., TURK G.: Rigid fluid: animating the interplay between rigid bodies and fluid. In *ACM SIGGRAPH 2004 Papers* (2004), SIGGRAPH '04, pp. 377–384. 2

[CT04] CHU N. S. H., TAI C.-L.: Real-time painting with an expressive virtual chinese brush. *IEEE Comput. Graph. Appl. 24*, 5 (Sept. 2004), 76–85. 2, 3

[CT05] CHU N. S.-H., TAI C.-L.: MoXi: real-time ink dispersion in absorbent paper. *ACM Trans. Graph. 24*, 3 (July 2005), 504–511. 1, 2, 3, 8, 9

[DKMI13] DIVERDI S., KRISHNASWAMY A., MECH R., ITO D.: Painting with polygons: A procedural watercolor engine. *IEEE Transactions on Visualization and Computer Graphics 19*, 5 (2013), 723–735. 2

[EWK*13] ECHEVARRIA J. I., WILENSKY G., KRISHNASWAMY A., KIM B., GUTIERREZ D.: Computational simulation of alternative photographic processes. *Computer Graphics Forum (Proc. EGSR 2013) 32*, 4 (2013). 2

[GSSP10] GOSWAMI P., SCHLEGEL P., SOLENTHALER B., PAJAROLA R.: Interactive SPH simulation and rendering on the GPU. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2010), SCA '10, pp. 55–64. 2, 3

[HBD10] HERAULT A., BILOTTA G., DALRYMPLE R.: SPH on GPU with CUDA. *Journal of Hydraulic Research 48* (2010), 74–79. 2, 3

[HK05] HONG J.-M., KIM C.-H.: Discontinuous fluids. In *ACM SIGGRAPH 2005 Papers* (2005), SIGGRAPH '05, pp. 915–920. 2

[Hoe12] HOETZLEIN R. C.: Fluids v.3: A large-scale, open source fluid simulator., 2012. URL: http://fluids3.com. 8

[JCM07] JIN X., CHEN S., MAO X.: Computer-generated marbling textures: A gpu-based design system. *IEEE Computer Graphics and Applications 27*, 2 (2007), 78–84. 2

[Kel06] KELAGER M.: Lagrangian fluid dynamics using smoothed particle hydrodynamics, 2006. 6

[LOG06] LEE S., OLSEN S. C., GOOCH B.: Interactive 3D fluid jet painting. In *Proceedings of the 4th international symposium on Non-photorealistic animation and rendering* (2006), NPAR '06, pp. 97–104. 1, 2, 3, 9

[MCG03] MÜLLER M., CHARYPAR D., GROSS M.: Particle-based fluid simulation for interactive applications. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2003), SCA '03, pp. 154–159. 6

[SN99] SAITO S., NAKAJIMA M.: 3d physics-based brush model for painting. In *ACM SIGGRAPH 99 Conference abstracts and applications* (1999), SIGGRAPH '99, pp. 226–. 3

[Str86] STRASSMANN S.: Hairy brushes. In *Proceedings of the 13th annual conference on Computer graphics and interactive techniques* (1986), SIGGRAPH '86, pp. 225–232. 6

[Suc01] SUCCI S.: *The Lattice Boltzmann Equation for Fluid Dynamics and Beyond*. Clarendon Press, Oxford, 2001. 3

[Wes91] WESTOVER L. A.: *SPLATTING: A Parallel, Feed-Forward Volume Rendering Algorithm*. Tech. rep., Chapel Hill, NC, USA, 1991. 7

[WW07] WANG C.-M., WANG R.-J.: Image-based color ink diffusion rendering. *IEEE Transactions on Visualization and Computer Graphics 13*, 2 (Mar. 2007), 235–246. 2

[WZF*04] WEI X., ZHAO Y., FAN Z., LI W., QIU F., YOAKUM-STOVER S., KAUFMAN A.: Lattice-based flow field modeling. *IEEE Transactions on Visualization and Computer Graphics 10* (2004), 719–729. 3