Energy-Brushes: Interactive Tools for Illustrating Stylized Elemental Dynamics

Jun Xing[1,2], Rubaiat Habib Kazi[1], Tovi Grossman[1], Li-Yi Wei[2], Jos Stam[1], George Fitzmaurice[1]

[1]Autodesk Research [2]University of Hong Kong

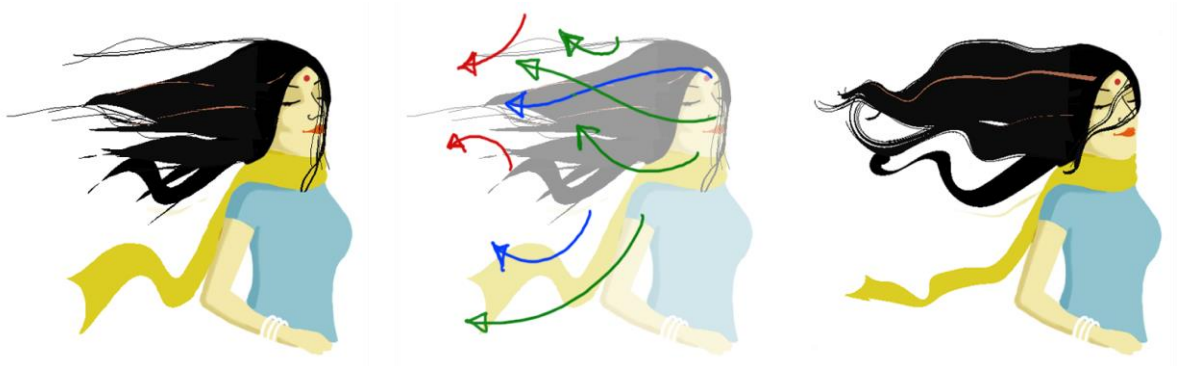

Figure 1: Example of a dynamic illustration authored with our system. Left: original user drawing. Middle: the *energy brush* gestures to specify the underlying forces and detail effects. Right: the resulting dynamic illustration (best viewed in Adobe Reader).

ABSTRACT

Dynamic effects such as waves, splashes, fire, smoke, and explosions are an integral part of stylized animations. However, such dynamics are challenging to produce, as manually sketching key-frames requires significant effort and artistic expertise while physical simulation tools lack sufficient expressiveness and user control. We present an interactive interface for designing these *elemental dynamics* for animated illustrations. Users draw with coarse-scale *energy brushes* which serve as control gestures to drive detailed *flow particles* which represent local velocity fields. These fields can convey both realistic and artistic effects based on user specification. This painting metaphor for creating elemental dynamics simplifies the process, providing artistic control, and preserves the fluidity of sketching. Our system is fast, stable, and intuitive. An initial user evaluation shows that even novice users with no prior animation experience can create intriguing dynamics using our system.

Author Keywords

Sketching; interactive illustrations; casual animation; dynamics.

ACM Classification Keywords

H.5.2. Information interfaces and presentation: User Interfaces – Graphical Interface.

UIST 2016, October 16 - 19, 2016, Tokyo, Japan

ACM 978-1-4503-4189-9/16/10…$15.00
DOI: http://dx.doi.org/10.1145/2984511.2984585

INTRODUCTION

"*As special effects animators, we do not animate things, we animate energy. Understand the energy behind the effect and stick to it with every stroke of the pencil*" – Gilland [9].

Dynamic special effect animations, such as waves, splashes, fire, smoke, hair, and explosions, are a sublime art form and an important aspect of animation. We refer this class of effects as *elemental dynamics*. In general, authoring these effects require mastery and deep understanding of the natural forces and elements. Despite considerable progress in the simulation of natural phenomena in computer graphics, user interaction and control remain challenging research problems. Traditional hand-drawn animation offers full artistic control and expression, but requires significant expertise and manual labor. Physical simulation can produce realistic animations, but can be hard to control and understand by non-experts especially for time-varying effects [2].

In recent years, researchers have explored sketch-based user interfaces [4, 12, 20, 25] for more accessible and controllable animation authoring. These works mainly focus on active and primary character animations and the resulting motions are typically *kinetic*, where the velocity itself is static over time. This makes these animations easy to specify and control. In contrast, the *elemental dynamics* are passive and secondary visual effects (VFX) where the underlying forces and motions change over time, making them challenging to control.

As illustrated in Gilland's seminal books on hand drawn special effects animation, "*Elemental Magic*" [8, 9], a classical hand-drawn approach to design these animation effects is to use *energy strokes* for implicit guidance

(Figure 2). Artists design special effects by sketching the gestures that coarsely define the underlying forces (e.g., heat ascension, air expansion, water drift). This approach enables artists to focus on the core underlying forces, exaggerate, stylize, and compose intriguing dynamics by the interaction of these forces, while preserving the fluidity of sketching. To the best of our knowledge, there is no interactive UI tool that leverages this approach as a way to design stylized animation effects.

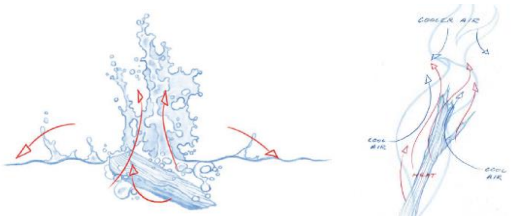

Figure 2: Artists use energy strokes (forces) to design, exaggerate, and stylize classical hand-drawn special effects animation. Image courtesy [8].

In this paper, we provide a new animation framework and interactive system that enable artists to design *elemental dynamics* by sketching the underlying forces with *energy brushes* to animate drawings and textures (Figure 1). An *energy brush* creates a stable, repetitive velocity field pattern by moving *flow particles* along the gesture. A *flow particle* creates a local velocity field that influences and deforms the neighboring shapes, creating the desired detailed effects. The framework is general and expressive, accommodating a wide range of effects - including water, hair, clothes, smoke, and other artistic ones, which are difficult or tedious to create otherwise. An initial user evaluation shows that even first time users with no prior experience to animation are able to create intriguing dynamics using *energy brushes*.

From an interaction standpoint, we contribute a novel user interface that balances between key-framing (with absolute artistic control, low physical realism, and high manual labor) and physical simulation (with low artistic control, high physical realism, and low manual labor). On the technical side, we contribute a novel method to approximate and automate traditional hand-drawn animation practices [8, 9] by a combination of *energy brushes* and *flow particles*, providing artistic control over coarse and fine scales. Our implementation is unconditionally stable [23] and runs in real-time, making it a suitable interface for dynamic illustrations.

MOTIVATION: ENERGY FOR ELEMENTAL DYNAMICS

"*In representing our effects through simplified versions of reality, we are free to focus our attention on the forces underlying the effects, the patterns in motion… Stylize, simplify, and find the energy focus!*" – Gilland [9]

Gilland's books [8, 9] repeatedly emphasized the use of energy (i.e., forces) as a means to design classical, stylized special effects animation. Numerous illustrative examples in the books portray how artists design stylized dynamic effects with primary, secondary, and often tertiary energy sources (Figure 2). The key idea behind the design of all these natural phenomena is *emergence*. *Emergence* is a process whereby intriguing dynamics arise through interactions among smaller and simpler energy patterns that themselves do not exhibit such properties. From a design perspective, the modularization of *stylized dynamics* into energy gestures is critical for several reasons. First and foremost, it helps to simplify complex dynamics into smaller, understandable chunks (i.e., energy). Second, in contrast to traditional CG tools based on simulation, it provides more room for stylization and control. Finally, from a conceptual point of view, it encourages the artists to focus on the core forces behind the effects, rather than the intricate details. In this paper, our goal is to design user interface controls for *elemental dynamics* that resonates with how traditional animators understand, learn, and execute such effects.

RELATED WORK

This section reviews prior works in the physical simulation of time-varying phenomena and sketch-based user interfaces for animation authoring.

Synthesizing Elemental Dynamics

A variety of natural phenomena (e.g., fire, water, smoke) can be simulated using fluids. In computer graphics, researchers have explored a number of techniques for fluid simulation [2]. Despite the progress in simulating these effects, their interaction and control still remain challenging research problems. In the context of stylized dynamic illustrations, the generality, expressiveness, and interactive control of the dynamics are of primary interest, compared to physical accuracy.

Key-framing

To facilitate artistic controls, key-frames are explored as constraints [5, 26] to solve for the appropriate forces to generate the resulting fluid simulation. Although key-framing provides full control to the animation, it requires significant expertise, and can be tedious when the animation is long. In contrast, this paper explores a complementary approach for authoring *elemental dynamics* inspired by the classical approach of sketching energy strokes.

Controlling detailed effects and coarse structures

Most elemental dynamic simulations require control over multiple scales – coarse and fine. The computer graphics community has explored various combinations of detailed effects and coarse phenomena, such as realistic fluid vortices [6, 22] and turbulences [15] over realistic coarse fluid flows; artistic flows [18], motifs [1], and mediums [3] over realistic coarse fluid flows; and realistic fluid flows over artistic coarse shapes [5, 26] or distributions [31]. In comparison, our design couples fine control via *flow particles* with coarse control through *energy brushes*.

Flow Particles

On the approach side, our *flow particles* idea is inspired by the use of particles to represent various phenomena in computer animation such as waves [32], vortices [22], and density motifs [1]. In theory, a given velocity field can be composed from *flow primitives* [28]. These flow primitives are based on physics, whereas our flow particles can represent both physical and artistic effects. Although *flow particles* are general enough to represent various local velocity fields, a key design issue is how to properly combine them with high level *energy brushes* as coarse controls to achieve desired visual effects and user expressiveness. This is a main focus of this paper.

Sketch-Based User Interfaces for Animation

The HCI and graphics communities have explored the use of sketching [4, 25], direct manipulation [11, 21], and auto-complete techniques [30] to make animation more accessible and easy-to-use. In motion sketching systems, the user sketches the motion paths [4, 25] or poses [10] to animate a given object. Draco [12] enables the animation of groups of similar objects, named kinetic textures, by providing motion controls over coarse and fine scales through sketching. Kitty [13] allows users to specify functional relationships between groups of animations produced by Draco. In Draco, the strokes are rigid and the velocity field remains invariant over time, which makes it challenging to produce dynamic effects like fire, water, clothes, and hair. Our work extends this line of work with a novel approach that provides the controllability of Draco, while exhibiting the time-varying dynamics of physical simulation, where the underlying velocity field changes over time and deforms the neighboring strokes.

More recently, Motion Amplifiers [14] leverages the language of traditional 2D animations by exposing the principles of 2D animation, which makes these dynamics accessible and easy, even to novice users. Motion amplifiers are designed and developed for active and primary 2D animations, while *energy brushes* address passive and secondary effects animations. In general, secondary effects are governed by external forces and energies, whereas primary character animations are governed by skeletal manipulations. In [14], animations are results of direct object manipulation. In our system, animations are first-order effects of the underlying velocity field, designed and specified by the artist.

In sum, these prior methods focus on different aspects of animations and can be combined with our framework.

OVERVIEW OF OUR ANIMATION FRAMEWORK

Based on our observation of classical hand-drawn *elemental dynamics* design [8, 9], we designed and developed an animation framework to design time-varying dynamics, while preserving expressiveness and fluidity of sketching. Our framework builds around general concepts that are easy to understand and use, while offering rich creative capabilities. In this section, we describe the core components of this conceptual framework. Later we describe our interactive user interface, and the technical implementation details.

Strokes

Strokes represent the hand-drawn lines and shapes, and imported textures. Each stroke is composed of a group of points, which can represent a polygonal line, a closed shape outline, a solid filled region, or an imported (and triangulated) texture, as shown in Figure 3.

Flow Particles

Compared to grid-based representations [23], we take an elemental approach [28] to model and control the underlying energy forces. *Flow particles* are the basic elements of our framework to generate a variety of detailed dynamics. A *flow particle* generates an energy pattern represented by a velocity field within a given radius. These flow particles move along *energy brush* trajectories to influence and deform nearby strokes, but not the motions of other flow particles to avoid infinite loops. The flow particles can be seen as a dynamic extension of kinematic templates [7].

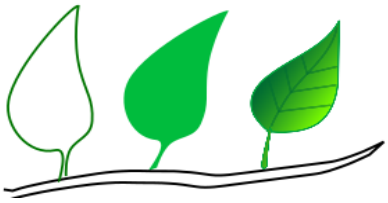

Figure 3: A stroke can indicate a shape outline (left), a region filled with solid color (middle), or a texture (right).

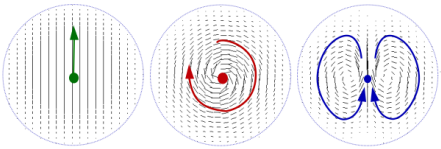

Figure 4: *Flow particles* with the wind, swirl, and smoke velocity fields.

Figure 4 shows three different types of *flow particles* – wind, swirl, and smoke - corresponding to some natural energy sources [8, 9]. A wind particle creates a velocity field that moves in a specific direction, a swirl particle creates a rotational velocity field that can spin nearby strokes, and a smoke particle generates an up-rising smoke-like motion. Apart from these pre-defined fields, users can also design custom, time-varying *flow particles*. The ability to create custom and time-varying *flow particles* makes our framework expressive, powerful, and potentially useful for a variety of stylized dynamics.

Energy Brushes

An *energy brush* is a stroke that defines the coarse direction of the energy and forces (Figure 5). Each *energy brush* is associated with a particular type of *flow particles*. It continuously emits the associated *flow particles* that traverse through the trajectory from one end to the other

end (Figure 5b). An *energy brush* creates a simple, stable, and periodic dynamic energy pattern. The brush specifies the coarse direction of the energy pattern, while the detail shapes are forged by the corresponding *flow particles*.

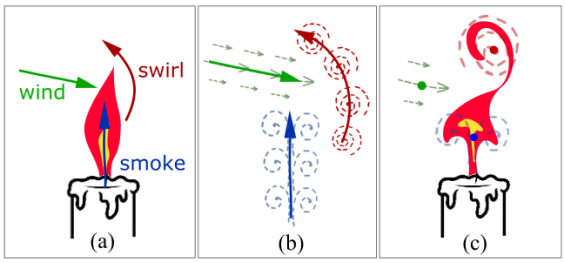

Figure 5: *Energy brushes*. (a) The colored arrows represent different types of *energy brushes* (*wind*, *swirl*, and *smoke*). (b) The *energy brushes* with corresponding *flow particles*. (c) The influence of energy brushes on a given stroke.

We consider four basic parameters for *energy brushes*: interval, speed, strength, and size, which correspond to the emission interval, the velocity of the *flow particles*, the strength of the *flow particle* velocity field, and the radius of *flow particle* velocity field, respectively (Figure 6). The spatial and temporal variation of these different parameters (Figure 6e and Figure 6f) can create a more dynamic and chaotic effect. Energy brushes can also be composed to create emergent phenomena (Figure 7).

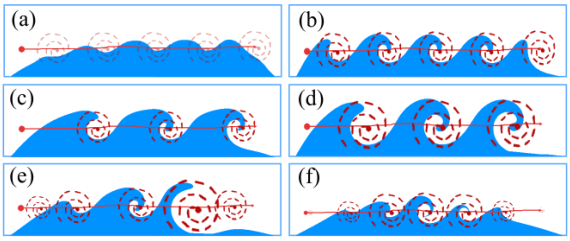

Figure 6: *Energy brush* controls: Varying the strength (a, b), interval (b, c) and size (c, d) parameters of the *energy brush* change the *flow particles* and their resulting effects. Furthermore, the generated flow particles can have parameters varying spatially (e) and temporally (f).

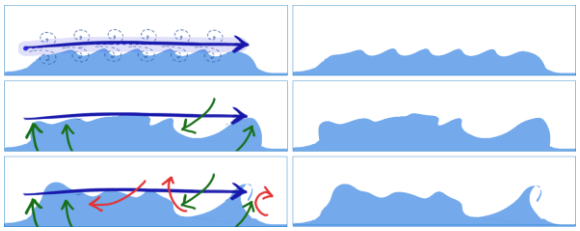

Figure 7: *Elemental dynamics* as emergent phenomena. Top: a single *energy brush* creates a simple, stable, and perfectly repetitive water design. Middle & bottom: subsequent *energy brushes* add randomness and details.

Constraints

Stroke constraints specify and control the stroke attributes that coarsely define their material properties. For instance, under the influence of an energy field, a hand-drawn smoke stroke would behave very differently than a stroke that represents a flag. The constraints define the shape attributes, avoid undesirable effects, and preserve structural details, as illustrated in Figure 8. An *anchor* constraint fixes the positions of a selected region of the stroke, a *stretch* constraint controls the length or flexibility of the stroke, and a *rigidity* constraint preserves the detailed shape in the selected region.

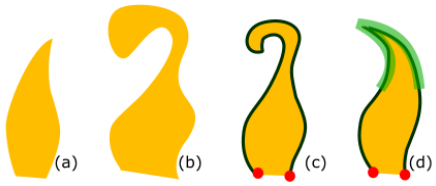

Figure 8: Stroke constraints. (a) The initial state of the stroke. (b) Without any constraint, the shape is not well preserved under the influence of *energy brushes*. (c) The *anchor* (red dots) and *stretch* (black outline) constraints control the position and length of the stroke. (d) The *rigidity* constraint (green stroke) preserves the shape (e.g. pointy edge) of the stroke.

USER INTREFACE

We designed and implemented our system to author stylized *elemental dynamics* in animated illustrations. Our interface builds on above animation framework, and capitalizes the fluidity and freeform nature of sketching for *elemental dynamics*. Figure 9 shows the user interface of our system, which includes a main canvas, a layer panel, a global toolbar, and a contextual toolbar.

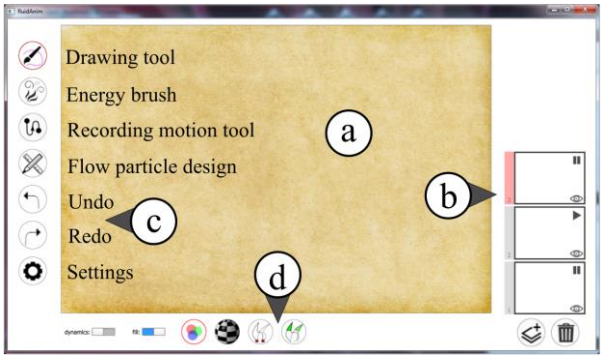

Figure 9: The user interface, consisting of (a) a main canvas, (b) layer panel, the (c) global and (d) contextual toolbars.

The global toolbar is comprised of the main tools for drawing, animating using energy brushes, recording motion, and designing new flow particles. Each of these main tools is associated with a contextual toolbar, which provides different control modes and feature options for the corresponding global tool.

In a typical workflow, the user starts by drawing the strokes to be animated. The user can then specify constraints (Figure 8) and animate the strokes using *energy brushes*. Multiple independent animation effects can be added to a scene by authoring them on separate layers. We elaborate on the workflow below.

Drawing

Creating Strokes

Once the *Drawing* tool is selected, the user can sketch strokes directly on the canvas using the brush tool. By clicking the brush tool, the user can open or close a widget panel which contains a color picker, brush thickness, and opacity slider.

The contextual toolbar for the drawing tool is illustrated in Figure 10. The user can specify whether the stroke is static or dynamic (Figure 10a). Static strokes are not influenced by the *energy brushes*. Users can also specify whether to fill the interior of the stroke (Figure 10b). The contextual toolbar also contains controls for adding textures (Figure 10c-d) and specifying local constraints (Figure 10e-f).

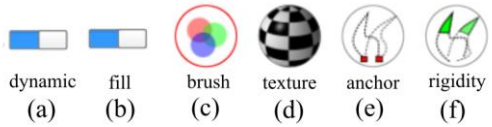

Figure 10: The Drawing contextual toolbar.

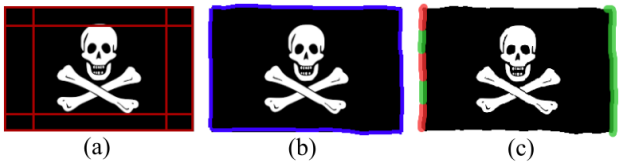

Figure 11: Load texture and add stroke constraints. (a) The user first loads an image to the main canvas and adjusts its size and position. (b) The user then lassoes the texture using the brush tool (blue). This creates a triangulated and textured stroke. (c) The user can specify the anchor (red) and rigidity (green) constraints by brushing over the edges.

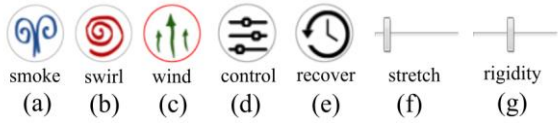

Figure 12: The *Energy Brush* contextual toolbar consists of the smoke (a), swirl (b), wind (c), and custom (not shown) *energy brushes*, a control widget to interactively edit the interval, speed, size, and strength parameters (d), a recovery tool to reset the dynamic strokes to their initial shape (e), and sliders to control the stretch (f) and rigidity (g) strength of the strokes.

Adding Textures

To add texture to a stroke, users click the texture button to load an image. The brush tool is then used to lasso a region of the imported image. This creates a new stroke that is filled with the loaded texture (Figure 11).

Defining Stroke Constraints

After drawing the strokes, the user can also specify local anchor and rigidity constraints to different parts of the strokes. The user selects the desired constraint type from the contextual menu, and then brushes over the strokes to add or remove the constraints. The anchor and rigidity constraints of the strokes are visualized with red or green color respectively (Figure 11c). The stretch constraint applies to the entire stroke (and thus no explicit icon under the contextual toolbar in Figure 10) and the user only needs to specify the strength (Figure 12f).

Energy Brush

Once created, users can animate the dynamic strokes by selecting the *Energy Brush* tool. This displays a contextual toolbar (Figure 12), which includes various types of *energy brushes*, a control panel to interactively modify their parameters, and global constraint sliders. A recovery tool resets an energy brush to its initial state.

Create Energy Brushes

To create an *energy brush*, the user first selects the *flow particle* type (i.e., wind, swirl, smoke) of the *energy brush*, and sketches the brush stroke directly on the main canvas. This creates an *energy brush* with a stable, repetitive energy pattern by continuously emitting corresponding *flow particles* along the brush stroke. The resulting brush influences and deforms the neighboring dynamic strokes immediately within the same layer. Each *energy brush* has a handle at its start point, and users can move or select the brush by dragging or clicking the handle. A brush can be deleted by dragging it out of the canvas. Intricate dynamics can be achieved by iteratively composing multiple *energy brushes* (Figure 13).

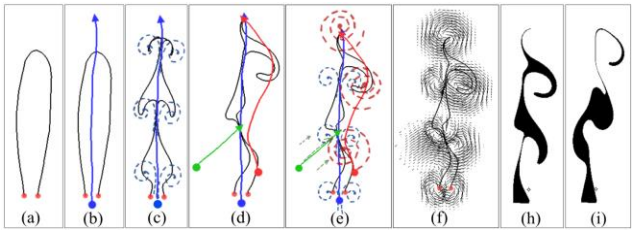

Figure 13: System Overview. The user first draws a stroke and adds anchors to its start and end points (a), then draws a smoke energy brush from bottom to up (blue) (b), which generates smoke *flow particles* to animate the stroke (c). The user can edit the animation by adding more energy brushes (swirl and wind) (d, e), and visualize the underlying velocity field generated by the flow particles (f). The user can also visualize the animated stroke as a solid filled region (h, i).

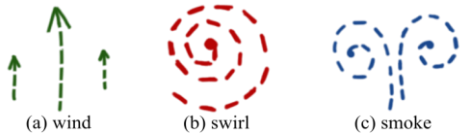

Figure 14. Iconic representations of different flow particles.

Visualizing Flow Particles

The *flow particles*, which travel along the trajectory defined by the *energy brush*, are visualized with an iconic representation of their effects (Figure 14). They are color-coded to identify and distinguish the type (wind, swirl, smoke). In addition, the size and opacity of the icon represent the size and strength of the flow particle.

Configuring Energy Brushes

Users can also select an *energy brush* and modify the interval, speed, size, and strength parameters (as visualized in Figure 6) using sliders in the control panel. The parameter changes are instantly applied to the *flow particles*, and the resulting dynamics are applied to the strokes with real-time feedback.

To instill some randomness into the dynamics, we also provide two types of variation controls in the control panel. For each control parameter, users can specify its degree of randomness. For example, by specifying 5% randomness to the size, the energy stroke will generate flow particles with slightly different sizes (Figure 6e). Users can also specify a temporal fade-in and fade-out effect for each control parameter. For example, the size of each flow particle can be gradually increased in the beginning and decreased in the end (Figure 6f).

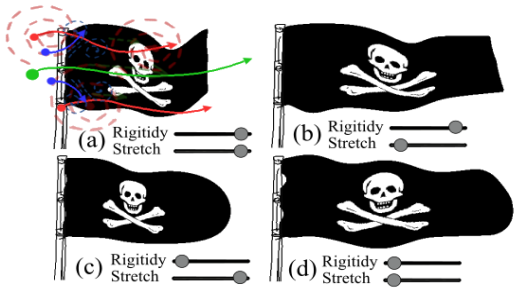

Figure 15: Constraints control. The user animates a flag with energy brushes (a), the resulting animations with different strengths of *stretch* and *rigidity* constraints (b-d).

Controlling Stroke Constraints

In the *energy brush* contextual toolbar, users can also globally control the strength of a stroke's *stretch* and *rigidity* constraints specified in Figure 11c. As shown in Figure 15, stronger *rigidity* and *stretch* constraints can preserve the local shape and overall length of strokes, while curbing the local and overall dynamics. This provides an easy and flexible way to emulate a variety of effects. For example, users can set weak constraints to create a dynamic smoke stroke (Figure 13), or a strong *stretch* constraint to create a waving flag (Figure 15a).

Designing New *Flow Particles*

To fully realize its potential and expressiveness, our system enables the users to design custom *flow particles*. Once user switch to the *Flow Particle Design* tool, they can specify the velocity field by sketching in the velocity field canvas (Figure 16a). This newly created flow particle is then displayed as an additional type on the *energy brush* contextual toolbar (Figure 12). Users can sketch *energy brushes* with the custom *flow particle*, and iteratively refine the velocity field to achieve the desired effect similar to what they do with other flow particles.

The design tool also allows users to design dynamic *flow particles*, in which the velocity field changes as the particle traverses through the brush stroke. A time-varying velocity field is composed of key-frames specified through multiple layers via the layer panel of the velocity field canvas (Figure 16b). The key-frames transition sequentially. Each key-frame has a defined lifetime to control how long it lasts during motion. For example, the user can design an explosion *flow particle* by specifying different emit directions in three different layers (Figure 16). When the explosion particles move along the *energy brush*, their emit directions change abruptly (short lifetime), resulting a drastic movement to neighboring strokes (Figure 17). Such drastic *elemental dynamics* would be very challenging to produce with existing static *flow particles*.

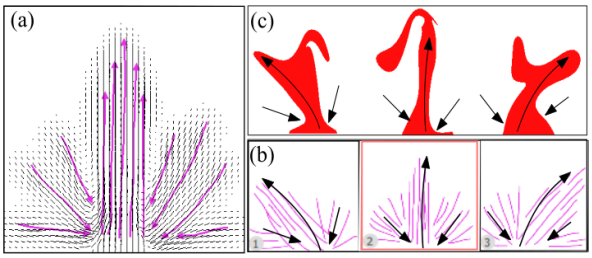

Figure 16: Flow particle design. (a) The user can design the velocity field by sketching (the magenta strokes) in the canvas. (b) By creating multiple layers of velocity fields, the user can compose a time-varying flow particle, which can generate more drastic-changing effects (c).

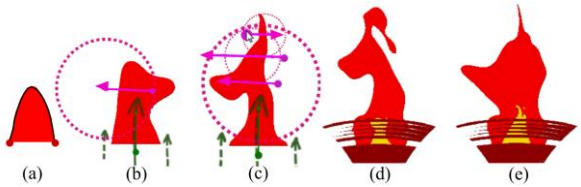

Figure 17: Example of fire animation with time-varying *flow particles*. The user (a) first draws a stroke filled with red color and adds anchors to both start and end points, (b) specifies a wind (green) and an explosion (magenta, designed in Figure 16) *energy brush* to create a coarse explosion effect, and (c) adds smaller explosion *energy brushes* on the top to create more fire details. The final effect is shown in (d-e).

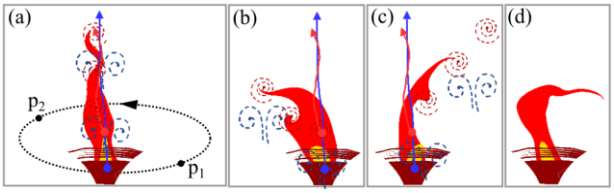

Figure 18: Motion scripting example. (a) The user specifies a motion path to an entire layer. The fire exhibits dragging behavior at point p_1 and p_2 due to the motion (b-c). Dragging is stronger with increased motion velocity (d).

Motion Scripting

With the motion scripting tool, users can record a translation path that applies to all the strokes and *energy brushes* in the corresponding layer. This facilitates the creation of layered animation effects. For instance, in Figure 18a the user creates a fire emitting from a static

torch. Scripting the motion of the torch base creates the effect of the fire trailing behind as the torch moves along the defined path. The resulting effect is obtained because the scripted path does not alter the trajectories of the already emitted flow particles (Figure 18b-c). Users can also adjust the motion speed to achieve different effects.

Layer Panel

We also provide a layer panel, analogous to sketching and graphical editing tools. Users can create and animate strokes in different layers independently from each other. Users can add, remove, and clean a layer, and can also control the visibility and animation state (e.g. run or stop) of each layer. This enables them to composite complex dynamic effects without interfering one another in an iterative manner.

SYSTEM IMPLEMENTATION

In this section, we describe the algorithm and implementation details of our system. In every frame, all the dynamic strokes are iteratively updated by the *flow particles* and the stroke constraints $c(s)$ (Figure 19). In order to maintain the stroke smoothness, we resample the strokes in each iteration. We elaborate each part below.

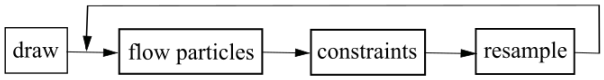

Figure 19: Algorithm overview. After being drawn, each stroke is iteratively updated by the *flow particles* and the stroke constraints. At the end of each iteration, each stroke is resampled to maintain smoothness.

Strokes

Analogous to [29], we sample each stroke with a constant distance (6-pixels in our implementation), and represent each sample s via a set of attributes:

$$(\boldsymbol{p}(\boldsymbol{s}), \boldsymbol{v}(\boldsymbol{s}), \boldsymbol{a}(\boldsymbol{s}), \boldsymbol{c}(\boldsymbol{s})).$$

This includes the position $p(s)$ and velocity $v(s)$ of s; the appearance parameters $a(s)$ such as color, size and texture coordinate; and the constraint parameters $c(s)$ that include anchor, stretch, and rigidity. To fill the region of a stroke, we triangulate the stroke samples via Delaunay triangulation. To add texture to a stroke, we map the texture boundaries into the corresponding lassoed brushes (Figure 11) via their texture coordinates.

Flow Particles

In our descriptions below, we denote p, v and p', v' as sample positions and velocities before and after being updated by the flow particles.

Representation

We represent each flow particle q as:

$$(p(q), v(q), F(q))$$

, which includes the position $p(q)$ and velocity $v(q)$ of q, and the local velocity field $F(q)$ defined in the local coordinate frame centered at q. We represent $F(q)$ as a *normalized* local velocity field $f(q)$ with tunable parameters $r(q)$ and $e(q)$ that control the size and strength of $F(q)$. This representation separates the design ($f(q)$) from the control ($r(q)$ and $e(q)$) of flow particles.

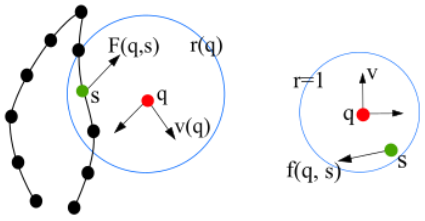

Figure 20: Flow particle representation. Left: the flow particle q (red) moves along $v(q)$ and influences all samples (e.g. the green dot) within its range $r(q)$ (blue circle). Right: the influence value $f(q, s)$ is calculated by mapping s to the normalized local velocity field $f(q)$.

Influence

We define the influence of a flow particle q to a sample s as $F(q, s)$, and its velocity $v(s)$ can be treated as the accumulated influences from all flow particles:

$$v(s) = \sum_i F(q_i, s) = \sum_i e(q_i)\, f(q_i, s)$$

, where $e(q_i)$ is the strength of q_i, and $f(q_i, s)$ is the corresponding velocity of $s$in $f(q_i)$, as illustrated in Figure 20. After calculating $v(s)$, the position of each sample s is updated separately by

$$p'(s) = p(s) + t_g * v(s)$$

, where t_g is the time-step, set to 0.1 in our implementation.

Different Flow Particles

Each normalized local velocity field $f(q)$ can be procedurally defined or represented by a grid of velocities. The wind, swirl, and smoke particles provided by our system are all procedural. Velocity grids are less compact but are general enough to represent any flows.

As illustrated in Figure 20, the range of $f(q)$ is 1, and the moving direction is $[0, 1]^T$. In our descriptions below, we assume $p(x, y)$ is a point within the range of $f(q)$, and $r = \sqrt[2]{x * x + y * y} \leq 1$ is the distance from $p(q)$ as the origin.

The wind particle (Figure 4a) is defined as:

$$f_{wind}(x, y) = (1 - r)\begin{bmatrix}0\\1\end{bmatrix}$$

, which simply creates the velocity field along the upward direction $[0, 1]^T$ and fading outward.

The swirl particle (Figure 4b) is defined as:

$$f_{swirl}(x, y) = (-1)^v \frac{1 - r}{r}\begin{bmatrix} y \\ -x \end{bmatrix}$$

, with a special case of defining $f_{swirl}(0, 0) = (0, 0)$ at the origin. f_{swirl} creates a rotational velocity field fading

outward, and $v \in \{0, 1\}$ is a parameter to control the rotation direction.

The smoke particle (Figure 4c) is defined as:

$$f_{smoke}(x, y) = \Phi(r) \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

, where $\Phi(r)$ is a 2×2 matrix defined as:

$$\Phi(r) = (\nabla\nabla^T - \nabla^2 \mathrm{I})\, e^{-r^2}.$$

$\nabla = \begin{bmatrix} \partial/\partial x \\ \partial/\partial y \end{bmatrix}$ is the gradient operator, $\nabla^2 = \nabla^T\nabla$ is the Laplacian operator, $\nabla\nabla^T$ is the Hessian operator, and I is the 2×2 identity matrix.

The velocity fields created by the swirl and smoke particles are incompressible (divergence-free), which guarantees against collision. It is also easy to see our wind particle will not cause collision. Thus, the combined final velocity fields will produce smooth animation without collision. For customized flow particles (e.g. Figure 16a), the velocity field can be made divergence-free via the projection operation [23].

Constraints

Anchor Constraint

If a sample s is applied as an anchor constraint, we simply fix its position all the time, i.e. $p'(s) = p(s)$.

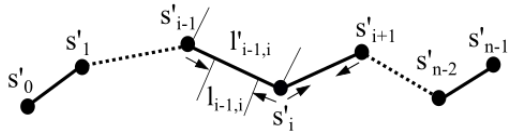

Figure 21: Stretch constraint. Each pair of adjacent samples is pulled towards or push against each other to maintain the original distances.

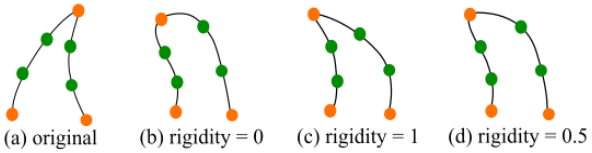

Figure 22: Rigidity constraint. (a) The original samples specified with rigidity constraint (the sharp angle in Figure 8a). (b-d) The results with different rigidity strengths.

Stretch Constraint

The stretch constraint is applied between all adjacent samples to prevent length changes. As shown in Figure 21, when the line between sample s_{i-1} and s_i is stretched or squashed, we move them towards or away-from each other by a distance of $0.5(\, l'_{i-1,i} - l_{i-1,i})$, where $l_{i-1,i}$ and $l'_{i-1,i}$ are the target and updated distance between s_{i-1} and s_i. Usually, the target distance is the original sampling distance between samples. For each stroke, this pulling/pushing operation is performed multiple times specified by a stretch parameter.

Rigidity Constraint

We use the as-rigid-as-possible deformation method [11] to constraint the shape of selected samples (e.g. the green and orange samples in Figure 22a). Specifically, we treat the samples on sharp turns or stroke ends (e.g. the orange samples in a) as control points, and the rest as free points (e.g. the green samples). Both control and free samples are first evolved by the flow particles (Figure 22b), then the free samples are deformed according to the updated control points (Figure 22c). However, the latter may lose the dynamics derived from flow particles, thus we provide a rigidity strength parameter to balance between the dynamics and shape preservation (Figure 22d) by interpolating the positions before (Figure 22b) and after (Figure 22c) the rigid deformation.

Resampling

When a stroke has no or only weak stretch constraints (e.g. a smoke filament), the distance between adjacent samples can become highly non-uniform after deforming along the flow particles, making the stroke unable to capture the detailed dynamics between the over-stretched adjacent samples (e.g. s'_8 and s'_9 in Figure 23b). To address this issue, we resample each stroke after being updated by the constraints.

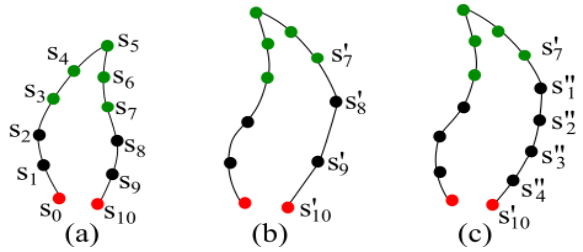

Figure 23: Stroke resample. When the stroke (a) gets stretched (b), we resample those parts (black dots) that are not anchor (red dots) or rigidity (green dots) constraints (c).

As shown in Figure 23a, each stroke can be divided into small segments based on the constraints specified by the user. The anchor/rigidity segments contain samples with only anchor/rigidity constraints (e.g. the red/green dots), and the free segments contain non-constraint samples (e.g. the black dots). The anchor segments are not resampled as they remain unchanged all the time. Since our rigidity constraint is based on the as-rigid-as-possible deformation [11] where the topology of samples remains unchanged, we also do not resample the rigidity segments. This is reasonable as the rigidity parts are less stretched due to the rigidity constraint. For the free segments, we resample them separately with the original sample distance. Due to the resampling, we also need to scale the target distance $l_{i-1,i}$ used in the stretch constraint accordingly. For example, to keep the overall length of the free segments $[s'_7, s'_{10}]$ in Figure 23b and Figure 23c, the target distance between adjacent samples in Figure 23c (5 pairs of adjacent samples) should be scaled to 60% of the target distance in Figure 23b (3 pairs of adjacent samples). Since resampling

may change the size of stroke samples, we re-triangulate each stroke after resampling.

INTIAL USER EVALUATION

We conducted an initial user study with both professional animators and illustrators, to gain qualitative feedback and insights on our tools. The study was also used to gather feedback on how our system contrasts existing approaches, but is not meant to serve as a formal comparison.

Participants

Six participants (5 males) took part in the study. Half of the participants (*P1*, *P3*, *P6*) were professional animators, and the remaining three were illustrators with little or no prior experience with animation tools. All the participants had moderate to good drawing skills. *P*1-5 were in-house, and *P*6 was external.

Study Protocol

All the experiments were conducted using a Wacom CINTIQ 21ux tablet display. The evaluation period lasted for approximately 75 minutes for each participant and consisted of the following steps:

Overview (20 minutes). Each participant was given a brief overview of the drawing and animation tools. The instructor walked the participant through a step-by-step tutorial to familiarize the participant with the system.

Exercise Task (15~20 minutes). In this step, participants were shown an animation consisting of two animated effects, water and cloth. They were asked to reproduce the given animations (Figure 24). In particular, the participants were asked to create a split effect in the center of the water dynamics, and were given the texture image to produce the animated cloth. No time limit was imposed and the facilitator did not intervene unless the participant had trouble completing the task. Our goal was to observe whether the participants could independently reproduce a target effect using our system.

Figure 24: Two frames of the exercise task. Each participant was given this background image, and was asked to create a water animation split in the middle plus animate the cloth.

Freeform Exploration and Feedback (30~40 minutes). In this step, the participants were free to explore the tools to create dynamic effects of their own. Finally, the participants filled out a questionnaire to provide feedback.

Results and Discussion

Overall, the participants found our system novel, playful, and easy to understand. After a brief overview and practice session, they were able to create stylized dynamic effects. Participants reported that compared to traditional, key-frame based animation tools, our system provided a more fluid (*P6*), faster, dynamic (*P4*, *P6*), and playful (*P1*, *P3*) interface to create dynamic effects. The real-time feedback of the dynamics was gratifying (*P2*, *P4*) and inspiring (*P2*, *P5*). Participants needed time to familiarize with our system but can gradually gain the experience to design more complex effects.

> *P2: "I like how fun and fast it is to see the result of the effects. It really begs you to play and experiment with it and it opens your mind and allow you to create effects that used to be difficult and painful to achieve."*

As compared to other animation tools for special effects, participants felt that the painting metaphor of dynamics design would make *elemental dynamics* more accessible to illustrators, artists, and designers (*P1*, *P2*, *P6*) across variety of application domains.

> *P3: "In Maya and other DCC's you can paint vector fields, draw motion paths for objects or forces and control object behavior with pinning and bend constraints. This tool rolls all of these multi-step actions into higher level tools with a brush-based paradigm."*

> *P6: "I can see this being used to bring life into pre-production drawings, animatics, web-comics, and children ebooks. You don't need to go to your special effects department to add these dynamics to your drawings."*

However, participants also reported that it took them a few minutes of exploration to comprehend the energy patterns, the emergent behavior of the brushes, and the resulting dynamics. Two participants (*P1*, *P6*) had difficulty in specifying the anchor and rigidity constraints when they were trying to create rigid character animations. The instructor assisted them to create their animations.

> *P2: "It was slightly confusing at first, but after a few minutes of playing with it, it was incredibly easy and intuitive."*

> *P6 "Very fluid FX such as water and wind look very good, but I found it difficult to create FX that would have a bit more rigidity."*

These observations provide important insights. The composite and emergent nature of *elemental dynamics* requires comprehension and understanding of the shape constraints, underlying forces, and their resulting behaviors. We believe that our tool reduces the barrier of creating stylized dynamics, but some level of learning and familiarization with the tool will still be required.

Exercise Task Performance

On average, the exercise task was completed in 9:10 minutes:seconds (min 6:50, max 15:10). All participants completed both the water and the cloth animation (Figure 24) without any assistance. The outcomes of the exercise task confirmed that our system can easily create simple dynamic effects.

Freeform Usage Observations

Overall, participants were satisfied with the range of effects that our system can support. Our participants authored a range of dynamic animations with our system in the freeform usage stage. *P2* animated a flag, smoke, and dust effect to illustrate a dynamic car (Figure 25). *P1* created a drawing with smoke effect (Figure 26) and an octopus with tentacles and hair floating in the water. *P6* created an animated jellyfish. Overall these examples represent a nice range of objects and materials that our system can be used to animate.

LIMITATION AND FUTURE WORK

While our system provides a general framework to produce, control, and stylize a wide range of elemental dynamics, it does not support (near) rigid object animations by Draco and Motion Amplifiers [12, 14]. Our system also cannot handle topology or color changes. Thus, it would be interesting to generalize the influence of *flow particles* beyond motion to other attributes (e.g. structure and color) to create more intriguing and stylized animation effects, such as breaking and merging of waves. In our system, severe distortion could poorly map the textures. To prevent this, one way is to increase the stretch constraint (weak distortion, e.g. flag), another way is to use sufficiently smooth textures. For severe distortion with complex textures, a volume-based representation (i.e., level sets, particles) might help, as our current implementation only simulates strokes but not areas.

The dynamics of our results are influenced by nearby flow particles. For complex dynamics, the influences of different flow particles may conflict and cancel each other. One way to mitigate this problem is to decompose a complex effect into layers (e.g. the hair example in Figure 1 with four overlapping hair strand layers) and hierarchies (e.g. creating the main motion followed by adding local variations). However, this requires some experience with the system. Customizing flow particles also requires experience, but once designed, it could be reused later. For example, users could use the explosion particle in Figure 16c to produce splash and explosion results.

Reproducing a given effect accurately can be difficult, as the resulting animations are first-order effects of the underlying energy fields. Thus, we use *similar* and *intriguing* as the quality criteria in our evaluation, as we consider these more important than high accuracy for stylized elemental dynamics.

Our participants also gave us some suggestions that could guide future enhancements. While the core forces (energy) in our tool are dynamic, participants (*P1*, *P3*) expressed their interests to have static forces that does not move or change over time (e.g., gravity). Together, the combination of static and dynamic energy forces could provide a powerful suite of tools for *elemental dynamics* design. *P3* was curious how this technique might apply in 3D environments, due to the inherent challenges associated to drawing 2D curves in 3D environment. Leveraging VR environments for sketching *elemental dynamics* in 3D might be an interesting avenue to explore. In our system, the *flow particles* are controlled by the *energy brushes* only. Making them context-aware [29, 32] may facilitate the creation of more complex dynamics. Our participants (*P2*, *P6*) also expressed their interests to specify the dynamics by directly drawing the strokes in the desired pose. This would require an optimization process to approximate the energy forces. In practice, such optimization techniques are very slow [5, 26]. This remains as a future work.

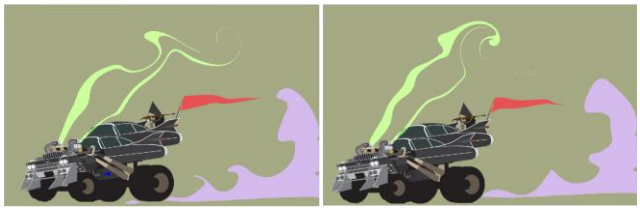

Figure 25: Two separate frames of the animated car created by P2 with dynamic flag (red), dust (purple), and smoke (light green).

Figure 26: Two frames of the smoker scene created by P1.

CONCLUSION

The basic philosophy behind classical hand-drawn *elemental dynamics* design is that complexity arises by combining simple energy patterns. Master animators approach *elemental dynamics* design by decomposing into simple, understandable components. User interfaces for animation that leverage these "simple components" as a natural vocabulary to understand and compose complex animations might have powerful influence on how animators think, learn, and explore [14]. In this paper, we have introduced a novel user interface for illustrating stylized *elemental dynamics*. Our main idea is to design the dynamics by sketching the underlying forces with *energy brushes* which serve as control gestures to move *flow particles.* Our animation framework approximates and automates classical approach of hand-drawn animation, provides artistic controls over coarse and fine scales, and preserves the fluidity of freeform sketching. Most importantly, it enables artists, as well as novices, to approach *elemental dynamics* design by composing them using simple energy patterns in an interactive and iterative manner, which is a complex process otherwise.

REFERENCES

1. Barnat, A., Li, Z., McCann, J., & Pollard, N. S. (2011, May). Mid-level smoke control for 2D animation. In *Proceedings of Graphics Interface 2011* (pp. 25-32). Canadian Human-Computer Communications Society.
2. Bridson, R. (2015). *Fluid simulation for computer graphics*. CRC Press.
3. Browning, M., Barnes, C., Ritter, S., & Finkelstein, A. (2014, August). Stylized keyframe animation of fluid simulations. In *Proceedings of the Workshop on Non-Photorealistic Animation and Rendering* (pp. 63-70).
4. Davis, R. C., Colwell, B., & Landay, J. A. (2008, April). K-sketch: a 'kinetic' sketch pad for novice animators. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 413-422).
5. Fattal, R., & Lischinski, D. (2004, August). Target-driven smoke animation. In *ACM Transactions on Graphics (TOG)* (Vol. 23, No. 3, pp. 441-448).
6. Fedkiw, R., Stam, J., & Jensen, H. W. (2001, August). Visual simulation of smoke. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (pp. 15-22). ACM.
7. Fung, R., Lank, E., Terry, M., & Latulipe, C. (2008, October). Kinematic templates: end-user tools for content-relative cursor manipulations. In UIST '08.
8. Gilland, J. 2012. Elemental Magic, Volume 2: The Technique of Special Effects Animation (Animation Masters Title). Focal Press.
9. Gilland, J. 2012. Elemental Magic: The Art of Special Effects Animation. Focal Press.
10. Guay, M., Ronfard, R., Gleicher, M., & Cani, M. P. (2015, June). Adding dynamics to sketch-based character animations. In *Proceedings of the workshop on Sketch-Based Interfaces and Modeling* (pp. 27-34). Eurographics Association.
11. Igarashi, T., Moscovich, T., & Hughes, J. F. (2005). As-rigid-as-possible shape manipulation. *ACM transactions on Graphics (TOG)*, *24*(3), 1134-1141.
12. Kazi, R. H., Chevalier, F., Grossman, T., Zhao, S., & Fitzmaurice, G. (2014, April). Draco: bringing life to illustrations with kinetic textures. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 351-360).
13. Kazi, R. H., Chevalier, F., Grossman, T., & Fitzmaurice, G. (2014, October). Kitty: sketching dynamic and interactive illustrations. UIST'14.
14. Kazi, R. H., Grossman, T., Umetani, N., Fitzmaurice, G. 2016. Motion Amplifiers: Sketching Dynamic Illustrations Using the Principles of 2D Animation. In CHI '16.
15. Kim, T., Thürey, N., James, D., & Gross, M. (2008, August). Wavelet turbulence for fluid simulation. In *ACM Transactions on Graphics (TOG)* (Vol. 27, No. 3, p. 50).
16. Ladický, L., Jeong, S., Solenthaler, B., Pollefeys, M., & Gross, M. (2015). Data-driven fluid simulations using regression forests. *ACM Transactions on Graphics (TOG)*, *34*(6), 199.
17. Lu, J., Barnes, C., DiVerdi, S., & Finkelstein, A. (2013). Realbrush: Painting with examples of physical media. *ACM Transactions on Graphics (TOG)*, *32*(4), 117.
18. Ma, C., Wei, L. Y., Guo, B., & Zhou, K. (2009, December). Motion field texture synthesis. In *ACM Transactions on Graphics (TOG)* (Vol. 28, No. 5, p. 110).
19. Müller, M., Chentanez, N., & Kim, T. Y. (2013). Real time dynamic fracture with volumetric approximate convex decompositions. *ACM Transactions on Graphics (TOG)*, *32*(4), 115.
20. Popović, J., Seitz, S. M., Erdmann, M., Popović, Z., & Witkin, A. (2000, July). Interactive manipulation of rigid body simulations. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques* (pp. 209-217).
21. Santosa, S., Chevalier, F., Balakrishnan, R., & Singh, K. (2013, April). Direct space-time trajectory control for visual media editing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 1149-1158).
22. Selle, A., Rasmussen, N., & Fedkiw, R. (2005, July). A vortex particle method for smoke, water and explosions. In *ACM Transactions on Graphics (TOG)* (Vol. 24, No. 3, pp. 910-914).
23. Stam, J. (1999, July). Stable fluids. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques* (pp. 121-128).
24. Takeo Igarashi, Satoshi Matsuoka, and Hidehiko Tanaka. 1999. Teddy: a sketching interface for 3D freeform design. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques* (SIGGRAPH '99), 409-416.
25. Thorne, M., Burke, D., & van de Panne, M. (2004). Motion Doodles: An Interface for Sketching Character Motion. *ACM transactions on graphics*, (3), 422-429.
26. Treuille, A., McNamara, A., Popović, Z., & Stam, J. (2003). Keyframe control of smoke simulations. *ACM Transactions on Graphics (TOG)*, *22*(3), 716-723.
27. von Funck, W., Theisel, H., & Seidel, H. P. (2006, July). Vector field based shape deformations. In *ACM Transactions on Graphics (TOG)* (Vol. 25, No. 3, pp. 1118-1125).
28. Wejchert, J., & Haumann, D. (1991, July). Animation aerodynamics. In *ACM SIGGRAPH Computer Graphics* (Vol. 25, No. 4, pp. 19-22). ACM.

29. Xing, J., Chen, H. T., & Wei, L. Y. (2014). Autocomplete painting repetitions. *ACM Transactions on Graphics (TOG)*, *33*(6), 172.
30. Xing, J., Wei, L. Y., Shiratori, T., & Yatani, K. (2015). Autocomplete hand-drawn animations. *ACM Transactions on Graphics (TOG)*, *34*(6), 169.
31. Yang, T., Chang, J., Ren, B., Lin, M. C., Zhang, J. J., & Hu, S. M. (2015). Fast multiple-fluid simulation using Helmholtz free energy. *ACM Transactions on Graphics (TOG)*, *34*(6), 2015.
32. Yuksel, C., House, D. H., & Keyser, J. (2007, August). Wave particles. In *ACM Transactions on Graphics (TOG)* (Vol. 26, No. 3, p. 99).