filtered.ink: Creating Dynamic Illustrations with SVG Filters

Tongyu Zhou
Brown University
Providence, Rhode Island, USA

Connie Liu
Brown University
Providence, Rhode Island, USA

Joshua Kong Yang
University of Massachusetts Amherst
Amherst, Massachusetts, USA

Jeff Huang
Brown University
Providence, Rhode Island, USA

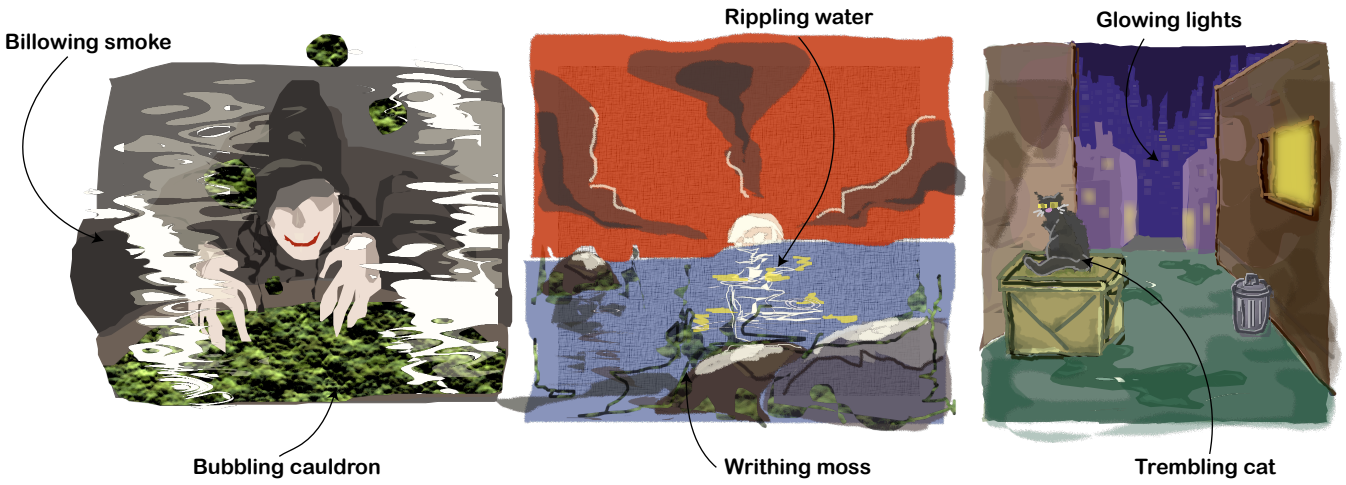

Figure 1: Example dynamic illustrations created using filtered.ink, with corresponding animated components annotated above.

ABSTRACT

Vector illustrations are object-based, meaning they are composed of strokes that can be filtered individually through textures or animations and transformed without loss of quality. These filters are typically difficult to specify without programming prerequisites. We propose filtered.ink, a full-featured illustration application to construct and explore filters via a node graph interface with a live preview. This turns vector graphics and their filters into a form of vector hypermedia that can be shared and remixed with new users. By examining interactions that occur when crafting, remixing, and using filters for dynamic illustrations through a task-based usability study, we expose new workflow patterns and avenues of expression. The observations result in a user model supported by filtered.ink: see, want, rewant, and remix. In this model, the artist breaks away from traditional notions of illustration, taking advantage of the inherent remixability of the strokes and filters in the vector graphics format.

CHI '23, April 23–28, 2023, Hamburg, Germany

ACM ISBN 978-1-4503-9421-5/23/04…$15.00
https://doi.org/10.1145/3544548.3581051

CCS CONCEPTS

• **Human-centered computing** → **User interface toolkits**; **Web-based interaction**.

KEYWORDS

vector illustration, SVG, animation, dynamic textures

ACM Reference Format:
Tongyu Zhou, Connie Liu, Joshua Kong Yang, and Jeff Huang. 2023. filtered.ink: Creating Dynamic Illustrations with SVG Filters. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems (CHI '23), April 23–28, 2023, Hamburg, Germany.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3544548.3581051

1 INTRODUCTION

Still images capture moments in time. When the static backgrounds in these images are juxtaposed against animated motifs, however, they are able to capture liminal *impressions* of time, where time is suspended in some elements yet flowing in others. This style of media, described as cinemagraphs [20] or kinetic textures [22, 23], allows artists to control the propagation of elements over time and lends to new creative possibilities. For clarity, we will collectively refer to this media within drawing as "dynamic illustrations."

Traditionally, GIFs and other video formats are used to create dynamic illustrations. While these raster formats have long dominated the web, they are not scalable to larger screens without incurring unwanted pixelation, which becomes a growing problem

with the rapid increase of screen resolutions. Their textures and animations also cannot be easily modified after export, as producing the resulting image file is an irreversible one-way conversion. Small changes to animation speed, lighting, or texture density in this traditional raster workflow thus become constrained and cannot be made to the underlying strokes. Conversely, vector-based media such as scalable vector graphics (SVGs) are pixel-agnostic and democratize the editing of dynamic illustrations, as the file format itself separates strokes from their modifying filters, each of which can be remixed into a new drawing (by remixing the illustration) or aesthetic (by remixing a filter). Unlike in raster, where edits are made sequentially and reverted from a linear undo stack, edits in vector can be *non-linear* by being applied to any past stroke. This ability aligns with previously expressed artist desires to save and apply different versions and styles of brushes more freely [25]. Non-linear editing can also facilitate rapid iteration and refinement of the illustration across different systems. These properties make vector graphics an ideal candidate for dynamic illustration.

Despite their benefits, vector graphics are not a common output format in illustration[1]. This paucity can be attributed to their limited forms of expression compared to those of raster-based illustrations. Although vector-based systems like Adobe Illustrator [16] and Inkscape [38] abstract away the complexities of vector programming through common drawing metaphors, they provide few to no options for creating detailed textures and animations. In particular, creating fluid animations for vector graphics has already been shown to be a non-trivial task for the user [3]. As a result, vector-based artwork is often limited to a minimalist, geometric appearance, restricting the variety of illustration styles an artist could express. We focus on this gap of texture and animation creation to enhance the breadth of vector illustrations.

To enable the authoring of dynamic vector-based illustrations, we create a web-based system called *filtered.ink* that abstracts stylization code for SVGs. Its node graph interface allows users to author animated computational brushes composed of SVG filters, or raster transformations that operate on existing vector primitives. The illustrations are re-rendered at each display resolution to empower users to create vector illustrations with the same expressiveness as that of raster. The platform provides several ways to create, manipulate, and remix these filter transformations that can then be applied onto a canvas. Since this system introduces an unfamiliar medium and new workflows, we then conduct a task-based usability study using filtered.ink with hobbyist and professional artists, who are familiar with common vector-based design tools but not SVG filters, to understand how the system can fit into the visual artist's natural workflow and influence dynamic illustration outcomes. By observing key behavioral patterns, preferences, and methods of expression unique to the vector format that are shared amongst the artists when using computational brushes, we identify a user model supported by filtered.ink and derive further insights into how vector-based creative tools with potential technical learning curves can be designed to better support the artist's natural mental model. From our study, we characterize an artist's workflow into four categories: see, want, rewant, and remix.

The contributions of this paper are 1) a full-featured system for dynamic vector illustration that enables users to achieve expressive raster-like styles through SVG filter authoring and 2) a user model for vector-based creative systems with learning curves informed by a task-based user study on workflow patterns and affordances.

[1]Although the exact numbers are hard to pinpoint, filtering for software that support vector output on ArtStation reveals vector graphics only occupy 171,500 out of 4.4 million digital 2D illustrations (3.9%). On sites dedicated to vector graphics such as IconScout, only 220,000 are illustrations or animations while 4.8 million are icons.

2 RELATED WORK

2.1 Vector Creation and Manipulation

Existing systems for vector-based media creation or manipulation fall into two general categories: human-centered approaches and automatic generation. One example in the former category is Sketch-n-Sketch, an SVG editor that demonstrates how direct manipulation interfaces with general-purpose programs can allow both experts and non-experts to use programmable systems [10]. This editor allows users to manipulate code to affect the rendered vector graphic and directly tweak the vector graphic to see the reflected changes in code, which the authors termed output-directed programming [11]. Other human-centered systems include commercially available tools like Adobe Illustrator [16] and Inkscape [38] for drawing, and Adobe After Effects [15] and SVGator [49] for animation. These systems support vector illustration and animation by abstracting away the programming aspects of vectors for physical drawing interactions and timeline-based animation frameworks, respectively. However, they separate the drawing and animation processes. In addition, Illustrator and Inkscape are drawing interfaces that do not provide easy ways to craft textures, SVGator is an animation tool but does not support the animation of textures, and After Effects only supports the dynamic masking or translation of static textures as animation. Similar to these systems, filtered.ink also relies on human-centered approaches for authoring. However, it extends existing approaches by providing a way to both directly create and animate the *compositions* of the textures as the user draws them. This synthesis of embedded animation mechanisms within a familiar drawing interface introduces a fluid end-to-end experience tailored for dynamic illustrations.

Using automated approaches can also result in rich vector-based media, albeit at the cost of user control. These examples include SVG-VAE [33], Im2Vec [40], Cloud2Curve [5], and Stylized Neural Painting [57], all of which rely on learned models to reconstructs vectors from an original raster graphic. For messier raster sketches with small spaces between sketched lines, Delaunay subdivision has been used for automatically coloring sub-regions [37]. Other methods operate solely within the vector space. DeepSVG [3] focuses on animating existing SVGs by producing smooth interpolations between one SVG and another, while temporal diffusion curves [32] introduces a new type of vector graphic that captures the evolution of strokes to achieve a wider variety of styles and effects for vectorized painting. These auto-generative approaches either excel in imitation but not in freedom of expression or suggest new system-specific file formats. filtered.ink also attempts to achieve raster-like levels of fidelity but incorporates greater user autonomy into the brush creation process.

2.2 Node-based End-user Authoring

Node-based editing of objects hails from historical prototypes such as Sketchpad by Sutherland [48], where a light pen manipulates the nodes that define vector-based shapes. More recently, it has been used in texture rendering algorithms to manipulate procedural material graphs. Some of these algorithms such as Hu et. al's pipeline [13] or MATch [44] train neural networks that require searching through pre-existing procedural node graphs to find a best match, while other strategies focus on semi-automatic generation to bypass the large data set requirement [14]. The resultant node graphs can be further fine-tuned by a user to change material properties such as roughness, color palette, or shininess.

Editing the generated materials and textures via a node graph is similar to the block-based approach in visual programming, which has been used in systems such as Scratch [41] and Blockly [9] to help novices understand the semantic connections between program state and output. In comparison to standard text-based programming, this approach has been demonstrated to incite greater learning gains and levels of interest in programming [54]. The visual metaphor of directly manipulating blocks is also particularly well suited for visual environments such as drawing and design. For example, Para [19] gives the user live manual control of declarative constraints to enable non-linear editing to produce complex procedural art. Similarly, Demystified Dynamic Brushes [30] links state to visual outputs so that users can control program execution through direct selection and Object-Oriented Drawing decomposes drawings into UI objects to be manipulated through direct touch [55]. While our work is similar to these end-user authoring systems by providing similar controls to produce visual outputs, we augment them by providing the additional ability to animate program states. These states are also preserved in the standalone SVG file that is exported from our system and can be rendered on all modern browsers and applications that support SMIL animation.

2.3 Dynamic Illustration

Motion within a static image, as opposed to short videos, can convey unique contextualized messages and emotions via dynamic visuals. To better understand the patterns that elicit these messages, previous studies have introduced design spaces for animated narratives [45] to provide useful guidelines to direct these animations. Other works have focused on improving ways vector-based animations in drawing are presented. For example, to avoid visual crowding when animations are scaled to smaller screen sizes, retargeting algorithms can preserve spatial detail and appropriately redistribute elements based on importance [43]. To better convey emotional expressivity, temporal flickering has been suggested. While flickering may seem like an undesirable artifact to some, adding *controllable* temporal flickering allows digital animation to mimic the raw quality of traditional, hand-colored ones [7].

Many authoring tools have also been created to help artists directly craft these dynamic illustrations. Draco [23] demonstrates the appeal of illustrations with kinetic textures and shows that continuous animation effects on standalone elements juxtaposed with still backgrounds can help designers create non-disruptive ambient motions. Kitty [22] pushes this further by introducing an underlying graph model to capture visual, spatial, and temporal relationships between drawn objects, and Druid [24] allows users to remix a set of motion amplifiers, based on preexisting 2D animation principles, for rapid iteration. In these systems, however, the user workflow involves sketching the static elements first and animating post-hoc by delineating motion paths or interaction patterns after. They also export to raster GIFs instead of vector graphics. filtered.ink is instead built on top of vector graphics and exports to SVG, thus enjoying the post export edit benefits of the vector format. While the resulting dynamic illustrations may be visually similar, our workflow also differs by enabling animation as a *first-class object*. As dynamic motion can be embedded as a filter transformation within the brush of the user, we support a workflow experience that extends traditional drawing methods with animation to provide minimal disruptions to an artist's natural drawing process.

3 FILTERED.INK

Our system design is informed by strategies for augmenting creativity through exploration and exemplars [28, 46, 47] and uses themes from existing tools for node-based authoring and dynamic illustration to empower interactions for artists to craft, manipulate, animate, and share SVG filters.

3.1 SVG Filters

Figure 2: Left: An SVG of a trump card without any filters applied. Middle: The SVG with a filter combining *feTurbulence*, *feDiffuseLighting*, and *feComposite* primitives applied to the background generate a paper-like texture. Right: The SVG with a filter combining *feTurbulence* and *feDisplacement* primitives applied to the diamonds to generate blur.

An SVG filter, also known as shaders in computer graphics, is a series of graphical transformations that can be applied to return bit-mapped results. It was first introduced in 2012 by the World Wide Web Consortium (W3C) to enable different rendering effects for SVGs, but can also be applied to any source graphic. The SVG filter output differs from that of traditional raster graphics as it is dynamically recomputed at each rendering resolution and thus the image retains high quality and fidelity. These filter transformations are defined with a `<filter>` tag, after which sets of elements called *primitives* with tunable parameter values can be sequentially composited to create different visual outputs, including imitating realistic textures or artistic styles (see Figure 2). With the current SVG 2 specifications, there are 17 possible filter primitives: *feBlend*, *feColorMatrix*, *feComponentTransfer*, *feComposite*, *feConvolveMatrix*, *feDiffuseLighting*, *feDisplacementMap*, *feDropShadow*, *feFlood*, *feGaussianBlur*, *feImage*, *feMerge*, *feMorphology*, *feOffset*, *feSpecularLighting*, *feTile*, and *feTurbulence* [35]. Multiple filters containing different permutations of primitives can also be defined within

the same SVG file and differentiated between via IDs. These filters can then be applied to SVG elements by adding an attribute `filter="url(#name-of-filter)"` to the SVG element. Currently, all major browsers support SVG filters and there are several online editors for SVG filters [2, 17]. However, these interfaces remain difficult to maneuver, do not apply the filters in an illustration context, and do not support animation.

3.2 Design Considerations

Informed by prior work, our vector-based drawing tool aims to augment the visual expressiveness of vector-based dynamic illustrations while preserving their benefits of infinite scalability, post-export texture and animation modification, and non-linear editing. Guided by two Shneiderman design principles for general creativity support tools [46], 1) support exploratory search and 2) enable collaboration, we explore the following two ideas in our system, targeting experienced artists but coding novices.

3.2.1 ***Learning curve via visual metaphors and abstraction.*** SVG filter authoring in existing vector drawing software involves either manually typing out the SVG filter code or tuning filter primitive parameters. To understand the benefits and challenges of both, we ran a qualitative pilot study on an earlier iteration of the system that employed both the manual code entry and parameter tuning strategies with 3 hobbyist digital artists. These artists were invited to explore both interfaces through 20 minute think-aloud sessions and interviewed about their experience afterwards. All three users preferred parameter tuning, but still felt that the learning curve was too high and not conducive to the exploration of different potential output effects. One user commented that filter-editing was "not visual enough" and thus was confusing to understand. This comment aligns with insights from previous work that called for developing efficient workflows that align with how artists *naturally worked* visually [18, 31]. Our final system thus incorporates greater visual abstractions and metaphors to better accommodate the artist's mental model.

3.2.2 ***Expressiveness via sharing and remix.*** Repeated exposure to other examples has been demonstrated to improve the quality of creative work through inspiration [28, 47]. Remix, defined as the process of taking existing artifacts created by someone else to combine or manipulate into new cultural blends [26], is an extension of this idea and has also been commonly used to generate new art. For example, in cases where the user is less familiar with animation techniques, crowd workers have been motivated to create re-mixable interactive behaviors and achieve "creative evolution" by inheriting, modifying, and adding features [29]. Previous studies into this combination of creative input from multiple generations of contributors have suggested that crowd-based design [56] is more effective than individually created ones. By supporting the sharing and remix of SVG filters, our system thus encourages users to interact with a multi-sourced continuous stream of references to incite greater varieties of output compositions.

As vector hypermedia such as dynamic SVGs are not commonly used in illustration, artist interactions with SVG-based, and by extension vector-based, drawing tools with learning curves have yet to be thoroughly explored. Understanding the workflow of how users create, manipulate, or remix these SVG filters using filtered.ink can reveal important insights into how subsequent similar systems can be further refined to accommodate evolving user mental models. We argue that this is a uniquely vector workflow, as modifying filters or illustrations from other artists directly from the output without any accompanying source files cannot be achieved in raster. Exploring the potential visual outputs this medium can produce through these workflows can also reveal potential directions for future development of additional visual effects. Thus, we use filtered.ink to explore the following two research questions:

- **Q1.** What are the natural workflows that users follow when creating illustrations with SVG filters?
- **Q2.** How do filters, as part of the structure of vector graphics, inform users' choices of style in dynamic vector illustration?

3.3 SVG Filter Editor

The SVG filter editor uses visual abstraction to organize different components of SVG code into distinct modules to encourage user exploration, as demonstrated in Figure 3. In this figure, artists can manipulate the underlying code, which is hidden unless they inspect the web DOM, through modules on the editor interface. These modules are A) the composition of SVG filter primitives, B) the tunable primitive parameters, and C) the animations that can be applied to each parameter. On this editor, the user can also see a live preview of the filter they are currently designing as they make changes, as well as delete or upload the filter to a global database to share with other creators.

3.3.1 ***Primitives.*** For simplification, since our tool focuses on drawing and only uses SVG polylines as the source graphic, we do not include *feImage* since it maps an input image to an output, *feOffset* since it only shifts an input, which can be achieved by drawing in a different location on the canvas, *feMerge* since similar effects can be achieved by modifying filter order, and *feTile* since it only tiles smaller inputs to an output. We include the remaining filter primitives to achieve pattern generation through Perlin noise (*feTurbulence*), pattern blending (*feBlend*, *feComposite*), pattern perturbation (*feMorphology*, *feDisplacementMap*, *feDropShadow*, *feBlur*), convolutions (*feConvolveMatrix*), color mapping (*feColorMatrix*, *feComponentTransfer*, *feFlood*) and lighting effects (*feDiffuseLight*, *feSpecularLight*). For these primitives, we use a node graph representation to depict the relationships between each filter in the composition, as illustrated in A of Figure 3.

Our node graph is a re-interpretation of the original stack-based structure of the SVG filter primitives. In the original structure, primitives are added sequentially and parsed in the reversed order that they were added. Certain primitives such as *feTurbulence* and *feFlood* that do not take inputs will override the effects of previous primitives. These overridden primitives remain in the filter code despite not contributing to visible effects and may incite confusion when users are inspecting the filter. To remedy this confusion, we visualize the filter stack as a node graph instead where each node represents a filter primitive and directed edges from some node A to a node B indicates that the result of primitive A is fed into primitive B as its input. The "active" set of primitives, or the primitives whose outputs are eventually fed into the last primitive in the stack, are

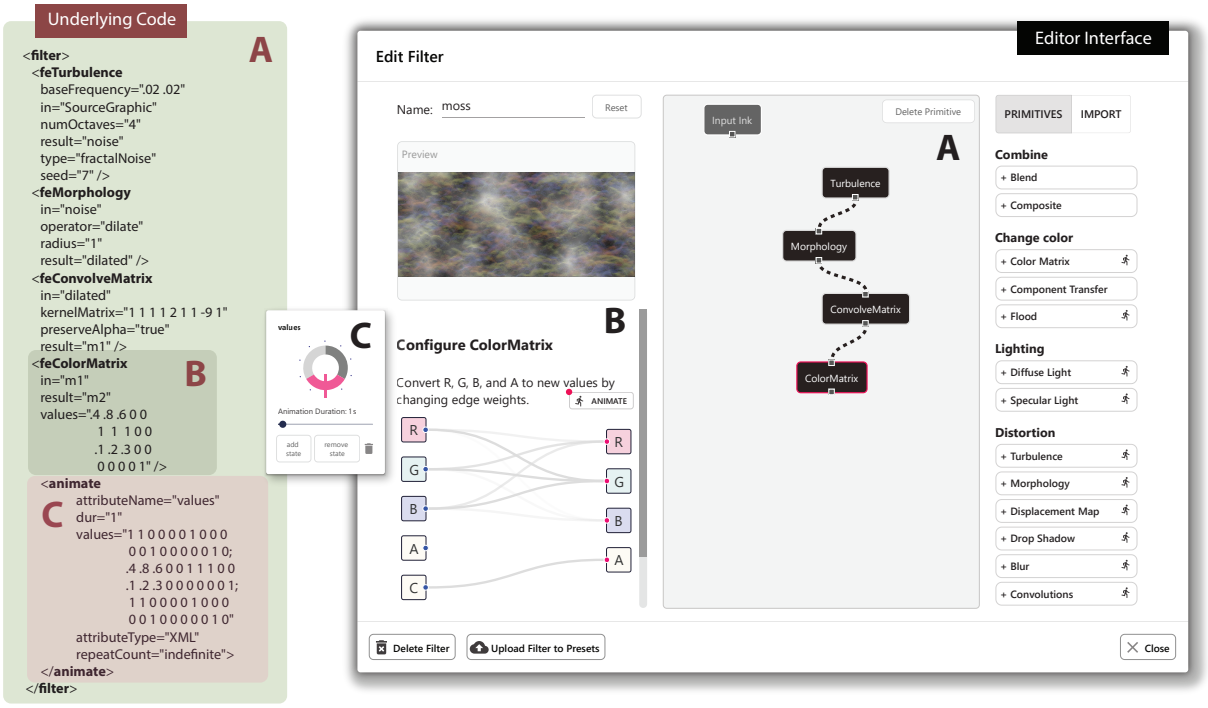

Figure 3: The underlying SVG code (left, highlighted in green) is abstracted through the filtered.ink editor interface (right). The editor interface uses A) a node graph representation to depict the connection between filter primitive inputs and outputs, B) visual abstraction to illustrate the operations performed by the filter, and C) a clock representing animation states for parameter values. In this example, values in the *feColorMatrix* filter are represented as edge weights in a bipartite graph.

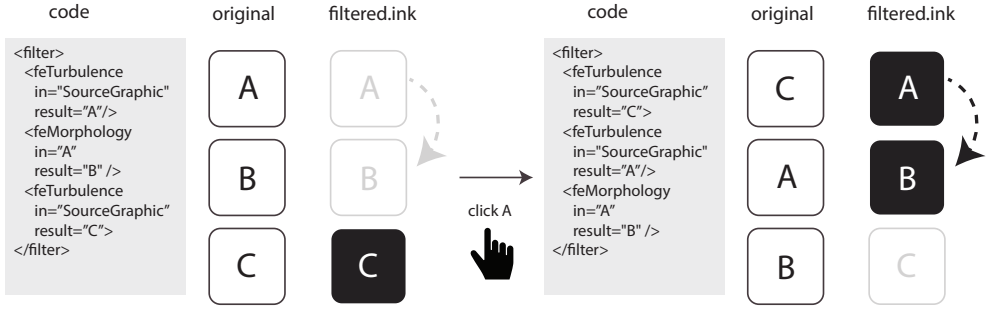

Figure 4: Left: The underlying abridged SVG code with the original stack data structure and the node graph data structure visualized by our interface. Primitive C is highlighted because it does not take an input so its output overwrites previous primitives. Right: When the user clicks on primitive A, the highlighted "active" set of primitives switch on the node graph. In the code, this involves moving primitive A and subsequent primitives associated with its output to the bottom of the stack.

highlighted in black while the overridden "inactive" set of primitives are rendered in light gray. The user can add, delete, and move nodes and edges to modify how the filters interact with each other as well as switch to a different active set of filters by clicking on the grayed out nodes, as demonstrated in Figure 4. This moves the clicked primitive with its subsequent associated primitives to the bottom of the stack to render their effects.

3.3.2 ***Primitive Parameters.*** After the user clicks on a node, the parameters of that primitive are visualized on a panel to the left of the node graph, as depicted in B of Figure 3. To better complement the visual workflow of the artist, we abstract away the actual code of the filter in place of visual representations. An example of one such abstraction can be seen in the figure, where the edge weights of a bipartite graph are used to represent matrix values in the *values* parameter of the *feColorMatrix* filter. The values in *feColorMatrix*

are used to apply a transformation matrix that converts existing RGBA values from the input to new ones in the output. Visually, the more opaque each edge is, the more one color channel in the original color scheme contributes to the color channels in the new color. Similar visual metaphors are applied to other filter primitives.

3.3.3 ***Animation.*** filtered.ink relies on W3C recommended SMIL 3.0 for filter parameter animations. Using this specification requires the user to provide the parameter to animate, the duration of the animation, and the values the animation cycles through. This type of authoring paradigm falls under the category of keyframe animation, where properties of objects are specified at certain points in time as keyframes, then frames between the keyframes are produced from tweening, or interpolating between the properties [50]. We select declarative keyframe-based animation over other paradigms such as procedural animation as prior work has demonstrated the former's ability to enable faster animation creation [27].

For simplicity, we restrict the animation to infinite loops through user-determined values. This is visualized in C of Figure 3. In accordance with the goal of visual metaphors, we abstract code away in favor of a colored clock divided into n chunks representing n values to cycle through for that parameter over the specified duration, similar to a timeline in standard keyframe-based animation. Clicking on each chunk populates the primitive parameter visualizer (B in Figure 3) with the values of that state, allowing the user to update parameter values using the familiar visual abstraction.

During editing, a live preview of the animation plays in the "Preview" section of the interface. A clock hand cycles through chunks with speed dependent on the user-specified animation duration for one completed cycle. Since the two are synced, the user can gain a notion of how changing parameter values directly influences the visual output, encouraging them to explore how parameter values of different filter primitives can generate a variety of visual effects. For example, animating the `scale` parameter in *feDisplacementMap* can make lines pulsate like rippling water. Animating the `values` parameter in *feColorMatrix* can make colors flicker.

3.4 Drawing Interface

The drawing interface, depicted in Figure 5, features a toolbar with common drawing tools such as pen, fill, color dropper, move, undo, redo, erase, import, and download, in addition to SVG-specific tools such as change filter. Users can draw directly on the canvas below the toolbar with their mouse, tablet, or by holding down the `D` key and moving their cursor. The sidebar on the left features the currently active SVG filters and the metadata of the most recently drawn or currently selected brush stroke. Importing a previously downloaded drawing will repopulate the sidebar with the filters used in that drawing. After users finish designing a filter, they can either directly start drawing with the filter applied as a brush or apply the filter to an existing SVG element on the drawing interface.

We enable cursor drawing by tracking mouse coordinates and mapping them to SVG polyline points. If a transformation is selected on the sidebar, we additionally pre-apply it to the stroke so that the effects can be seen as the stroke is being drawn. Each time a new stroke, or polyline, is drawn, we store the coordinates, color, width, opacity, time the stroke started, time the stroke ended, its associated SVG filter transformation ID, and a unique creator ID. To optimize performance, we simplify the polyline using a combination of the Douglas–Peucker and Radial Distance algorithms [1].

3.5 Remix Affordances

As remix has been previously demonstrated to improve creative outcomes by providing inspiration and encouraging learning by example [28, 47], we were interested in examining remix behaviors specific to SVG filters. Thus, we added support for three types of remix: 1) remixing filter parameters, 2) remixing filter primitives, and 3) remixing the filters that are applied to strokes, as demonstrated in Figure 6. After selecting a filter from a list of presets, the user can either remix the filter directly or click on a "Remix" button, which clones it before opening up the filter editor. Then, they can make small adjustments via parameter tuning or rearrange the filter primitive node graph to make visually larger disparate effects. They can choose to upload these transformed filters as new "presets" to a global database, by clicking on the "Upload Filter to Presets" button in the filter editor. Other users on the application can then browse through these preset filters, import them, and remix them for personal use. Users can also remix the filters applied to strokes on the drawing interface. To help with this decision-making process, we include a gallery of entire drawings with pre-applied filters so users can import them to see which filters are used in tandem with what kinds of drawn objects. They can then replicate or remix this idea into their own illustrations.

4 EVALUATION

We conduct a usage evaluation for filtered.ink to 1) understand the natural workflow of how users create dynamic illustrations and 2) examine how filter usage may inform, affect, or modify the styles of vector illustration. We also assess the application's conceptual clarity, usability, possibilities, and limitations. By examining the interactions and creative potentials of users when drawing with their own crafted filters as brushes, we can derive insights into designing future iterations of vector-based illustration applications.

4.1 Participants

Nine participants took part in the study (6 female, 1 male, 2 nonbinary), aged 20 to 22 years old (average 21). All participants had been creating art in some capacity since childhood and actively produce digital artwork for either a publication, professional practice, schoolwork, or recreation. Three were studying illustration in university. Four participants had experience with coding, while five had introductory or no experience at all. Participants were recruited from the researcher's own artistic network, as well as from illustration teams from digital campus publications. The participants were all students from two neighboring universities, one of which focuses on art. Eight of the nine participants listed Procreate, Photoshop, and Illustrator as the applications they currently use to draw with, with the former two used predominantly for digital painting and the latter for design. Every participant listed PNG as their usual file format of choice for their artwork. No participants had any prior experience or knowledge about SVG filters. Each participant received a cash compensation of $25 for their participation.

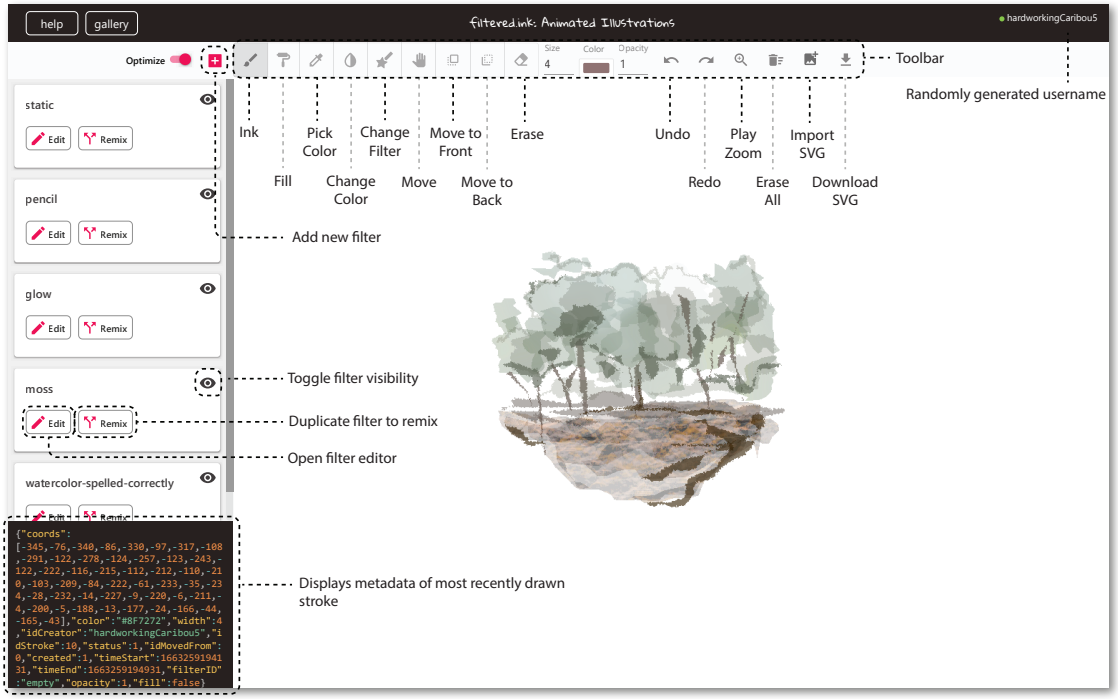

Figure 5: The drawing interface features a toolbar with common drawing tools, the canvas, a list of active SVG filters, and metadata about the most recently drawn stroke.

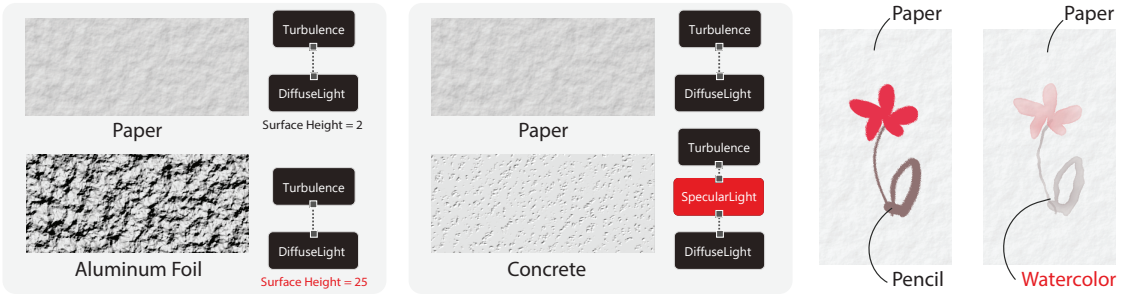

Figure 6: Examples of the three types of remix enabled by our system. Users can 1) create an aluminum foil filter from paper by increasing the surface height parameter of the DiffuseLight primitive, 2) create a concrete filter from paper by adding a SpecularLight primitive, or 3) create a different style by replacing the pencil filter on the flower with a watercolor filter.

4.2 Study Protocol

We perform our evaluation via a task-based usability study with semi-structured interviews for user modeling, and record interaction logs to capture habits users adopt when creating vector illustrations. The study was conducted using a Huion H420 Pen Tablet, mechanical keyboard, and 1920x1080 monitor. Each participant was asked to bring into the study an existing illustration they'd made on another digital platform to encourage user focus on the creative medium instead of devoting attention to the subject matter of the illustration. Informed by prior work that found learning instances to be greater in specific tasks in comparison to open-ended exploration [42], we constrained the participants to tasks that asked them to recreate existing filters and illustrations (rather than creating from scratch) to focus on participants' abilities to understand and integrate filters into their existing workflow and art styles. The evaluation period lasted 80 minutes for each participant, and consisted of the following steps.

4.2.1 ***Introduction (5–10 minutes).*** Participants filled out a background questionnaire about their experience with art, coding, and vector graphics. Each participant was then given a brief

overview and demonstration of the application, including its gallery of preset filters and completed drawings. The facilitator walked each participant through the interface and the functionalities available on the platform, demonstrating steps such as selecting preset filters, creating a new filter, adding primitives, and customizing effects. Participants were encouraged to try out the interface for themselves and get comfortable with the setup, as well as narrate their thought processes out loud.

4.2.2 ***Task 1: Filter Creation (15–20 minutes).*** Participants were asked to select an existing preset filter they liked from the list of pre-made filters and recreate it. The exercise covered the discovery of a desired visual effect, the creation of a filter from scratch, the procedures and learnability of adding primitives, customization, and feasibility of mimicking that desired effect. Participants were prompted with the preview of the target preset filter on the preset drop-down list as well as with their set of active filters on the filter sidebar, which they could refer to and utilize at any moment. They were prohibited from entering the preset filter's editor to view how the preset had been created. The facilitator did not intervene unless the participant had trouble using the system, and a time limit of 20 minutes was imposed. The purpose of this task was to observe whether the participants could easily reproduce a target effect.

4.2.3 ***Task 2: Free-form Drawing (35–45 minutes).*** Participants were asked to recreate the illustration that they had brought in using filtered.ink. Participants were encouraged to utilize and integrate filters into their workflow and artwork in whatever way they saw fit or was intuitive to them. Once done with their artwork, participants were asked to fill out a post-study questionnaire to provide feedback about the application. After the questionnaire, we conducted a semi-structured interview to understand the artists' thought processes and their overall experience of the system.

5 RESULTS

5.1 Supporting Natural User Workflow

We map a new user's natural creative workflow via interaction logs recorded throughout the tasks, as depicted in Figure 7. In this plot, "drawing" interactions refer to instances whenever a participant drew strokes on the canvas, while "filter editing" interactions refer to instances whenever a participant changed filter parameter values or edited filter animations. Filter editing primarily occurred in the beginning during the filter creation task. However, when the filter creation task concluded, most participants continued to edit filters during the free-form drawing task, where they had more freedom to work in ways more intuitive to them. After an uninterrupted filter editing phase, participants changed their workflow to focus on predominantly drawing on the canvas. However, most participants still returned to the editor to make short bursts, or clusters, of filter modifications. These observations imply that there could be a shift of mental models from creative to technical and vice versa that occur when artists switch between the two tasks.

5.1.1 ***Designing Filters.*** As they were designing filters for the first time, all participants initially relied on blind experimentation when attempting to understand what each filter primitive could do and how to achieve their desired effects. They would employ trial

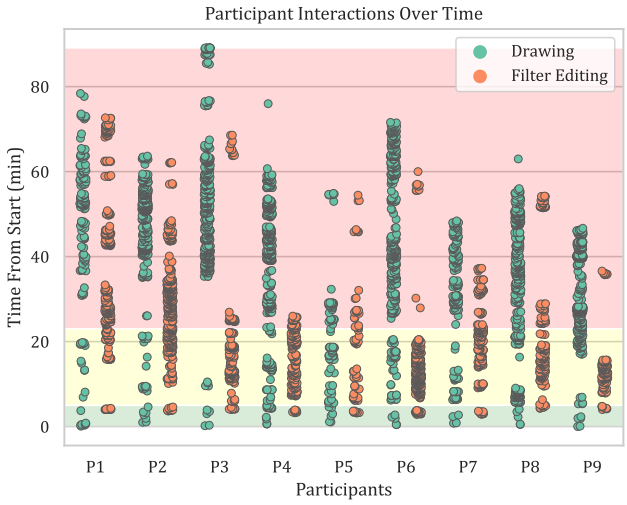

Figure 7: Instances where participants were drawing and editing the filters. The colored backgrounds refer to the average time taken for each task (green: introduction, yellow: task 1, red: task 2). Filter editing generally occurred in a large phase near the beginning, after a bit of drawing but before a large amount of drawing. But filters were lightly edited during the drawing for most participants.

and error, clicking on each primitive to preview its visual output before application, then repeating this process with different combinations of primitives and primitive parameters. When comparing this exploration process against those in similar existing systems, P7 stated that "the learning curve of this is much lower than other vector art applications." However, participants still found certain visual abstractions to be more intuitive than other ones. In particular, most found the *feConvolveMatrix* primitive intimidating to create from scratch. To accommodate for these spikes in difficulty, there was a notable reliance on remixing existing filters without facilitator prompting, especially for participants with less coding backgrounds. P3 explained, "I don't know what a lot of the primitives mean, so the presets were easier to use and alter, rather than start from zero." The provided presets on filtered.ink thus became an integral resource, both as a list of examples to demonstrate what the platform is capable of and as a palette of starting points for users to customize to their own needs without having to make something from scratch. P2 related this feature to her own artistic practices on existing drawing platforms, "I always like to use preset brushes. So much time and thought goes into making them." To remix these presets, *all* participants started by tuning the filter parameter values to make minor adjustments. While some were satisfied enough with the results and moved on to drawing, others continued to add filter primitives to the node graph and experimented with different connections between different nodes. Generally, those who continued to this second type of remix (P1, P2) used greater numbers of total remixed filters in their subsequent illustration.

In the beginning, if the filter was originally static, participants usually added animations *after* they were completely satisfied with

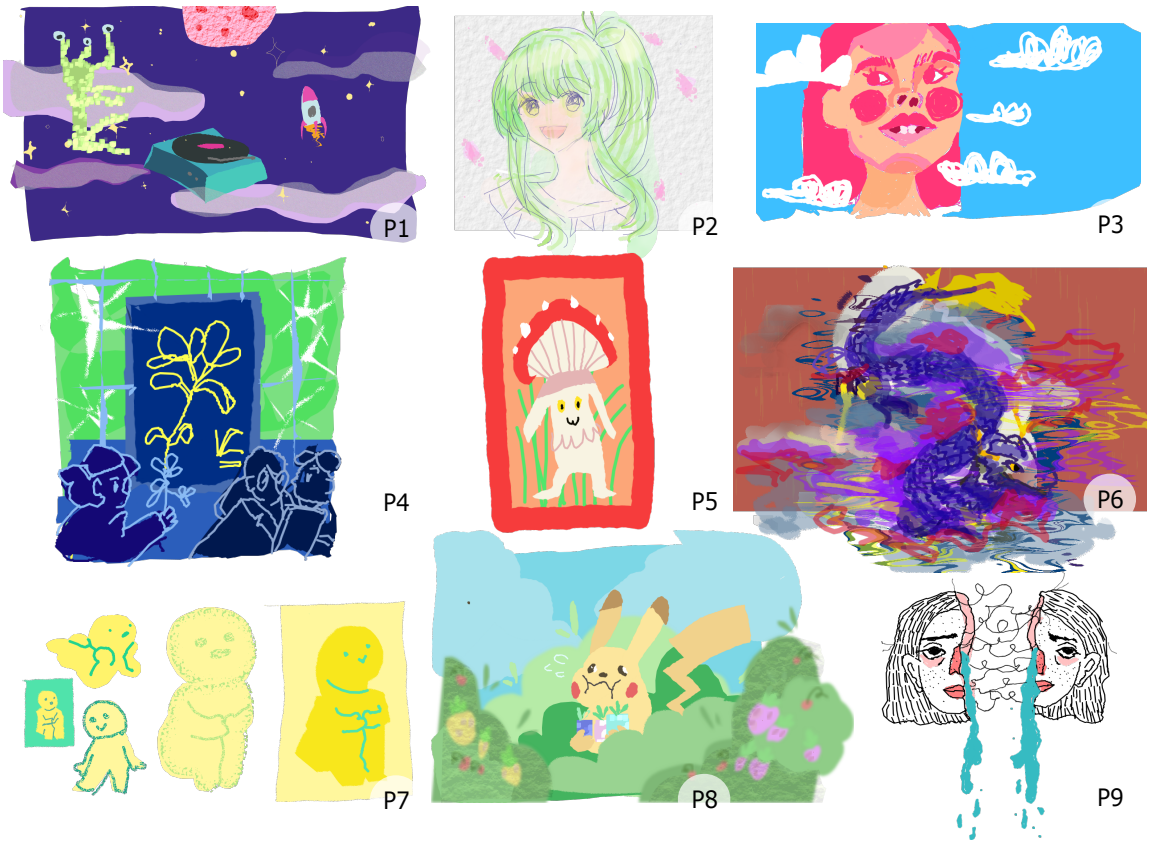

Figure 8: Final participant illustrations created using filtered.ink. Some participants used filters to emphasize environmental elements (e.g., rain and fog in P6), while others used them to create subject emphasis (e.g., tears in P9).

the static version of the filter and did not touch non-animation related elements again. Some participants who had less experience due to the "inaccessibility and big learning curve of existing animation tools" (P5) felt that filtered.ink's simple clock-based looping animation worked as a gateway into animation, that "it's easier than key-framing and adding a Gaussian Blur and exporting it to a GIF." To understand what is happening, participants stated that they would shift their gazes back and forth between the rotating hand that passed over different primitive parameter states and the live preview to match parameters with their direct visual outputs. In particular, P6 stated that this made them more confident to experiment, "It's more intuitive to use initially. If you're an illustrator without a background in animation for web design, this is really helpful. There's no added burden of having to learn how to code." These participants, after greater familiarity with what the animation function does, were comfortable enough to alternate between editing the static and animated aspects of the filter. During the subsequent semi-structured interview, all participants confirmed animation to be one of the greatest strengths of filtered.ink and what set it apart from all existing drawing applications. P7 also mentioned the potential of animated vectors to be "great for use in texturing for 3D models and movie applications," pointing to the potential to recycle math-based animations for game assets. P5 was excited about using animated SVGs in illustrative typography, suggesting, "instead of having to export each animated letter form as a GIF, having an animated SVG would save much more space and be infinitely resizable for every publishing platform."

5.1.2 ***Drawing with Filters.*** When drawing, participants relied heavily on features that matched those of existing creativity support tools. P7 specifically stated that, "I'm imparting my knowledge of Photoshop and Procreate onto this" when navigating both the filter editor and the toolbar on the drawing canvas. They immediately understood what common functions such as the eyedropper, fill, erase, and color tools did, and tried out the SVG-specific functions that they were less familiar with first. Participants also pointed out their desire for additional functionality that have become standard on most other popular drawing platforms such as resizing, polygonal shape tools, and a more robust selection tool. The lack of these small functionalities was cited as the biggest inhibitor to them immediately adopting filtered.ink into their existing workflows.

Participants fell into two categories of filter integration in their drawing process. Some participants first drew a skeleton of their illustration with standard non-filtered brushes, and then applied filters afterwards either with filter brushes or by using the "Change Filter" tool to convert existing non-filtered strokes into filtered strokes. We noted that these participants were not always satisfied

with their first choice of filters, and instead toggled between 3-4 others before deciding on a final one. Other participants drew directly with the filters themselves. Participants that first drew with non-filtered strokes mentioned that it was difficult placing an animated stroke where they wanted it to be given its movement, "so it was better to draw first and then click and choose what part I wanted to be animated" (P9). Participants that drew with filters directly felt that it was more important to see the end result as they drew, "It's useful to just draw with the pizzazz itself" (P6). These participants seemed more intentional with their filter choices and did not remix the filters applied to strokes as often as the previous group. Either way, both groups of participants were working with frameworks of existing drawing platforms that they were accustomed to in mind; those who preferred static, unadorned strokes usually work with print or comics, while those who preferred animated, textured strokes create animations and other forms of dynamic digital art.

Two participants also naturally gravitated towards the "Import SVG" function, expressing that this would be really beneficial to current digital creators and communities that already work a lot with the vector graphics but do not have tools to generate textures or animations. These participants would upload the filters they designed so that others could re-adapt them in their own artworks. Other participants who did not upload stated that they felt the filters they crafted weren't "unique" enough, but were willing to upload if they designed one they were satisfied with. However, both expressed concerns about whether the platforms, browsers, and mediums that they publish work on would be compatible with displaying SVGs. Many online publications that they illustrate for only accept standard raster image files like PNGs and JPGs, and they wished SVGs were more universally used.

5.2 Filter Effects on Aesthetic Style

The participants' final drawings can be seen in Figure 8. Despite limited experience working with vector graphics and no knowledge of SVG filters prior to using the system, all of the participating artists were able to incorporate filters to recreate their illustration and enhance it in a novel way. Seven of the nine participants, when given the opportunity in the last free-form activity to draw and edit filters freely, either remixed or created new filters that they liked enough to incorporate into their final drawing compositions. A breakdown of the filter distributions per drawing is in Figure 9.

We decompose each participant drawing into strokes and map their distributions by filter type in the first graph of Figure 9. All 9 participants incorporated *animated* filters into their drawings, while only 2 participants incorporated *static* filters. This observation is reiterated by P4, "Animations are the coolest part, because the static effects can be achieved using existing brushes on other applications." Since the participants were familiar with raster-based tools for creating visual effects and patterns in drawing, they stated that the ability to animate rapidly in-situ was the most impressive thing about the system. This finding indicates that although artists are cognizant of the latent benefits of vector graphics such as scalability, visual benefits seem to be more greatly valued.

Drawings with more than 200 strokes have a smaller percentage of strokes filtered ($N = 5, \overline{x} = 14.4\%, sd = 5.5\%$) in comparison to drawings with less than 200 strokes ($N = 4, \overline{x} = 59.3\%, sd = 29.7\%$).

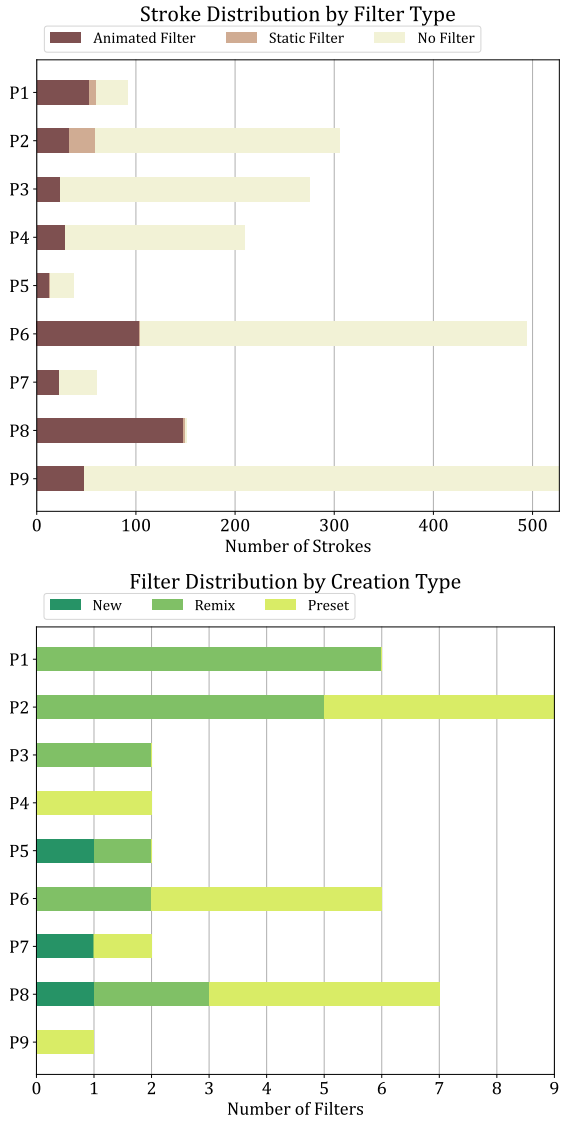

Figure 9: The first graph shows strokes separated by filter type. A higher percentage of strokes are filtered in drawings with less than 200 strokes total in comparison to drawings with more than 200 strokes total. The second graph shows filters separated by how the filter was created. Most participants remixed filters while only two stuck with preset filters.

In the former more complex drawings, animated filters are used sparingly to highlight particular facial features (hair in P2, blush in P3, tears in P9) or to bring environmental elements to life (flower petals in P2, clouds in P3, rain and fog in P6). This style resembles that of the cinemagraphs in Cliplets [20] and the kinetic textures of Draco [23]. Conversely, in the latter drawings with less strokes, the filters are evenly distributed across different objects in the scene and used to convey the characters of these objects. For example, in P1's illustration, the entire body of the alien receives a stringy, pulsating

filter alluding to its strangeness, the moon receives a rough filter to mimic its porous texture, the clouds receive a static-like filter that slowly moves up and down to evoke the feeling of floating, and the fire of the rocket receives a wispy filter that resembles smoke trails. This style is more reminiscent of the scenes from "Loving Vincent," which uses a laborious paint-on-glass animation technique [52].

The second graph of Figure 9 contains a breakdown of the filters in each drawing by how they were created. Two participants (P4, P9) only used preset filters while two other participants (P1, P3) only relied on filters they remixed from the presets. Interestingly, the participants who solely relied on presets also ended up with comparatively more complex drawings, or ones with more strokes. This correlation could imply that they were not yet comfortable enough with the filters in general to incorporate them into the composition. The rest of the participants employed a mixture of new filters they crafted from scratch, remixed filters, and preset filters. Given the time allocation of 30-35 minutes, participants who designed their own filters from scratch were only able to create at most one new filter for their drawing. These participants also had final drawings with comparatively less strokes, indicating that creating these filters from scratch was time consuming.

6 DISCUSSION

Vector-based authoring and dynamic illustration tools traditionally separate the workflows for drawing [11, 16, 38] and texturing/animation [3, 17, 49]. We demonstrate that instead of keeping them distinct, synthesizing the two processes within the context of a single drawn brush stroke through the "filtered ink" metaphor provides a compelling and fluid experience for the user. Some participants attributed this enjoyment to animation, "the fact that it [the SVG stroke] could move on its own is so cute. It looks like it's living" (P8). Others liked "the ability to texture vector images. Because the current process is so tiring, you don't see that around in a lot of vector art" (P7) and described the authoring style as "nostalgic" (P3), which they noted was rare in vector-based software.

Informed by existing block-based authoring tools that helped novices draw connections between technical program states and visual outputs [9, 19, 41], we primarily relied on node graph manipulation and parameter tuning in our filter editor. This representation was successful, as participants with a primarily illustration and little programming background found the application to be an accessible introduction into creating vector-based dynamic illustrations, with one participant noting their willingness to continue to use it post-study to incorporate more vector art into their current portfolio. However, the entirety of the learning curve could not be mitigated, as filtered.ink, similar to other systems within the design space of computational brushes and dynamic illustrations [22, 23, 25, 30], coerces the artist to deviate from their pre-established traditional workflows with its introduction of textures and animations that can modify strokes. There is no clear "grammar" describing how these elements interact and could be authored. Thus, we identified patterns in how participants interacted with the unfamiliar medium of an animate-able computational brush during our user study in an attempt to create this grammar. While these observations were made within the scope of a node graph technical drawing tool, we believe they are useful in informing the experiences of similar vector-based or technical creative systems that expose live code or programs, and summarize them into a four-part user model: 1) Users are motivated to explore and use a new system by seeing immediate inspirational examples (*see*). 2) They want to create something, but still rely on knowledge from prior mediums (*want*). 3) As they explore the system, they adjust their mental models to accommodate new possibilities (*rewant*). 4) To actuate new visions, users mix familiar techniques and outcomes with new ones created by others using the system (*remix*). We additionally provide examples and implications of each stage and summarize them in context within other studied artist-technical tool relationships.

6.1 *See*: Artist motivations and actions are guided by immediate inspirational visual stimuli.

Participant choices with filtered.ink were directed by recent visual examples that were personally appealing to them. For example, when selecting presets to use and recreate in Task 1, they mostly gravitated towards the animated ones: "The animation part is the best feature and what makes it unique; I work with print mostly. It could be useful for graphic design people." (P9) This suggests that the unique ability of SVG filters to enable both the creation and fluid *transfer* of motion from one drawn motif to the next was most motivating for users. Similar sentiments were shared by other participants, and this influence was reflected in the artworks they chose to create. At their own discretion, all the participants either directly included their recreated filter or a variation of it in their final illustrations. Outside of animation, some participants were attracted to the style afforded by the textures from filters, which reintroduced traditional raster-like aesthetics to the vector space. P5 mentioned that "It's interesting that it's a very painterly tool which isn't the case with other vector tools" and this novelty motivated them to more deeply explore filter combinations that could portray similar effects in their illustration.

The benefits of visual examples for providing inspiration have been previously explored [21, 28, 47, 53] and calls have been made for tools to facilitate the retrieval, storage, and dissemination of examples [12]. Many existing SVG tools also feature examples within external galleries or demos to motivate the user through exemplars that demonstrate the possibilities of vector [16, 38, 49]. However, as the immediacy at which individual users experience these community-curated, and thus unstructured, examples vary, we emphasize situating interactions with inspirational examples directly within the workflow of the creative tool if immediate action is desired. We note that immediate visual stimuli may lead to direct imitation and fixation, however, commonly presented as the antithesis of creativity. We argue that in this instance, fixation is not necessarily undesirable and can contribute to a greater user motivation to immediately jump into a technical creative tool.

6.2 *Want*: To achieve novel desired visual outputs with new tools, artists initially still rely on familiar traditional methods.

Participants initially retained their familiar mental models when interacting with the system. Most participants (P1, P2, P3, P4, P5, P8,

P9) preferred the traditional approach of drawing in plain strokes then applying animated or textured SVG filters post-hoc instead of directly drawing with them. They cited the reason as familiarity with their original workflow of drawing and animating/post-processing separately. This separation is complemented by the node graph workflow as designing stacking SVG filter primitives with clear inputs and outputs visually reinforced the idea that filters are embellishments applied to strokes. Participants described that drawing with motion is distracting, with P4 stating that "it's clearer to draw a shape down and apply a filter to it. [Animated] filters are a bit difficult to see the edges of." This reliance on traditional methods is similarly reflected in their mental models of SVG or vector-based elements. P1 stated, "Filters would be a secondary add-on because the artistic concept comes first," implying that textures and animations are not natural parts of the artistic concept. This sentiment is reiterated by P2, who compared filters to "the icing rather than the cake" and is consistent with the traditional practices in digital art of using line art to reduce forms into their most simplistic representations before adding embellishments. Without any modifications to the strokes, viewers are able to focus on the mass and volume of the subject. In this sense, although an SVG-based drawing system introduced a way to draw animations directly, most artists did not initially take advantage of this due to the interaction functioning as the antithesis of familiar line art. However, there were two participants (P6, P7) who preferred drawing directly with animated brushes. When prompted, these participants cited the same reason provided by those who preferred plain brushes, familiarity. P7 attributed their inclination to their usual workflow in existing digital illustration platforms, where when they select a computational brush and draw directly with it, the brush's resulting stroke is the final result. P6 additionally explained why they also preferred filtered brushes, "it's difficult to visualize what the final result will look like without drawing directly with the filter. Especially with the water filter, it looks so different from its original stroke underneath." Despite following contrasting workflows, the two groups of participants both demonstrated initial reliance on strategies that aligned with familiar mental models.

Other creative systems with learning curves can similarly complement the introduction of new tooling with the ability to attach familiar workflows. This attachment can be encouraged by providing familiar visual metaphors such as a white screen representing a drawing interface [18, 22, 23] or familiar interactions such as direct manipulation instead of using proxies [10, 19, 39]. In instances when what is familiar may be unclear, such as our observations with filter application, the flexibility to support multiple potential workflows can provide comfort when the user is learning new techniques without alienating any particular user base.

6.3 *Rewant*: After greater familiarity with the tool, artists construct new mental models.

Once the user familiarizes themselves with the new creative tool, established patterns and mental models may gradually change. Previously, some participants remixed the static parameters of preset filters before working on animation edits, considering drawing and animation to be separate processes. After longer usage of filtered.ink, however, they began to alternate between the two as the line between drawing and animation blurred. Some participants specifically reported that the ability to craft animated filters allowed them to think about the notion of time for individual objects and devise new strategies for storytelling. The motion of "sunlight rays that move and radiate" (P2) can be slow to portray the mood of a lethargic afternoon or flash rapidly to reflect the mindscape of someone running a marathon in the scathing sun. This mentality also shows the participant starting to think of lighting as a non-destructive layer for the scene, noting later that the lighting can change based on "if you want a scene to be morning day and light."

Our SVG-based paradigm presented filters as dynamic brushes. Participants mentioned that working under this metaphor emphasized thinking about the roles of brush texture and animation within storytelling *during* the drawing process and consequently encouraged them to re-imagine the kinds of scenes they could create. Under this new mental model of "re-imagined space and time" by controlling the animation of specific elements, they created drawings that could be divided into two categories: animations for 1) emphasis or 2) mood. Drawings that employed animation for emphasis (P2, P3, P6, P9) apply filters to one or two objects that the participants wanted to stand out. This choice created an illusion of foreground objects moving in a loop juxtaposed against static backgrounds or moving background objects against a static foreground, a style used in cinemagraphs that can afford "monstrative pleasure transcending any narrative quality of the image" [4]. Conversely, in drawings that employed animation for mood (P1, P4, P5, P7, P8), filters were applied more liberally to a large majority of objects, both foreground and background. This choice resulted in a different style of illustration where each object had its own unique "character" attributed by the animation. Elements with the same filter animations applied meld into the same cohesive unit and adopted this same "character."

Similar reconstructions of mental models have been historically repeated with the introduction of creative tools. The most drastic is the transition from traditional to digital mediums, enabling artists to rethink drawn strokes as maneuverable layers. We also see similar changes in tools like Mental Canvas [6], where users adding depth to images start to ideate in 3D while drawing in 2D. Another example is Cliplets [20], where users creating cinemagraphs from static photos found that they started to focus on small, isolated motions in their everyday lives and see the world within these microcosms. Similar to our case, the observed shifts in these systems were not intended, but rather discovered as side effects. Thus, more conscious decisions from system designers on the potential ways new interfaces may alter artists' pre-established thinking patterns can result in better designed infrastructures to support the creation of personally poignant media.

6.4 *Remix*: Artist final creations are facilitated by the synthesis of personally preferred results with multiple experimental styles.

We observed several patterns in which participants remixed the SVG filter presets: remix of filter parameters occurred before the remix of filter primitives, the remixing order of static and animated elements were separate initially but became intermixed with changing mental models, and those applying filters to plain SVG strokes

were less deliberate with remix decisions than those drawing directly with filters. Participants reported two main benefits of this remix process: to 1) help them understand which combinations of filter primitives can enable what effects and to 2) incorporate other styles into their own illustration. By starting with an existing node graph and tweaking internal parameters, they were able to use trial and error to parse through differences in values and establish mental models of what each primitive can do. This finding is consistent with prior work that reveals how people can learn complex software through trial-and-error [34]. Since there is no strict "error" in the trial-and-error process of parameter-based creative authoring, only subjective notions of more "attractive" or "favorable" outcomes to denote success cases, the ability to save instances of remix can not only help the user learn the technical details involved but also delineate and parse out the operations necessary to achieve divergent visual outcomes and share these with others. The increased understanding of filters and depth of filter exploration, as a result of our participants' successive iterations of each other's brushes and their additions to the list of presets, are consistent with prior findings that suggest the benefit of multiple generations of contributors in creative processes [56].

In prior works, remix occurred at varying degrees of abstraction, ranging from lower complexities of basic customization to higher complexities of medley remix [8], both of which have their drawbacks and benefits. Lower complexity systems such as Thingiverse's Customizer that enable parameter manipulation of object properties [36] are user-friendly but produce results that can be traced back to the original parent. Conversely, higher complexity systems like Sketch-a-bit where artists add strokes to build upon a shared motif [51] require greater knowledge of the system but generate results where the original parents are unrecognizable. We observed that filtered.ink provided different complexities to different participants; some only twiddled with the parameters and preserved the filter's visual forms while others completely rearranged filter primitives such that the original filter was unrecognizable. Those who chose to upload their remixed filters as new presets for others to see when presented with the opportunity fell in the latter category. For similar creative tools with remix capabilities, emphasizing high complexity remix within the export and import work cycle can potentially increase the likelihood of users sharing their remixed examples. This sharing is pivotal to the generation of new visual examples, feeding back to the first stage of this user model to inspire the exploration of others. What's unique about the nature of vector graphics is that both these filters and strokes can be remixed in their final output form. They are stored separately in the SVG files, enabling the full cycle of remixing from the output of another dynamic illustration. This is unlike GIF or PNG where remixing of the output is inherently impossible due to the assimilated blending that occurs when the work is converted to its final form, even if it had remixable elements while being composed, e.g. in prior tools with dynamic elements and programmable brushes [20, 23, 30].

6.5 User Model Summary

Using filtered.ink, we wanted to explore the natural workflows that users followed when illustrating with SVG filters (Q1) and how these filters influenced the styles of dynamic vector illustration (Q2). We found that these questions are closely intertwined with our proposed user model. Visual examples of filters provided by the system incentivized the user to explore the interface through fixation on a preferred style within the examples. Influences of the fixated style were transferred to certain strokes in the final illustration. To create this illustration, users initially relied on familiar workflows, although the definition of "familiar" differed based on prior experiences with static and dynamic mediums. Those in the former category used filters to post-process strokes, while those in the latter category drew with animated textures directly. As the users gained greater comfort, they reconsidered pre-established mental models for illustration and started thinking about how to use particular animated filters to associate mood, character, and emphasis with individual elements in the drawing for storytelling. To achieve their final results, they relied on remix at varying degrees of complexity to combine their ideas with filters created by others. If they felt the remixed results were distinctive enough, they opted to upload them to serve as examples to others.

This user model is one that stems from filtered.ink's SVG and vector-based affordances. Filtered strokes were *seen* as inspirational and motivating because users operated on the basis that the textures and animations can be freely edited and transferred non-linearly without losing quality across iterations of exported drawings. When participants *wanted* to recreate these effects, they were able to apply familiar mental models that kept drawing and post-processing independent using a node graph interface that presented the filters as separate stacking effects. As they continued to use the filter-as-brush metaphor which incentivized the synthesis of drawing with animation/texture application, however, they felt more inclined to *rewant* the stories associated with each drawn element. Finally, *remix* of the SVG can be performed and shared at various levels of breakdown of the drawing (filter primitive parameters, filter primitive combinations, stroke, and drawing), lending to greater creative possibilities in web-based collaborative work.

While its inception is rooted in SVGs, the user model can also be applied to other systems for dynamic illustrations or those that explore relationships between artists and technical tools. For example, in Draco [23], artists *wanted* to create ambient motion with respect to individual objects, but usage of the tool coerced them to *rewant* and think in terms of more general animated patterns instead. Findings in Demystified Dynamic Brushes [30] similarly revealed that artists *wanted* to rely on familiar interactions initially, then "an hour into the session I felt like I was translating between the world of numbers and artwork" [30]. These artists also relied heavily on changes they could *see* to direct their manual manipulation of program state and underlying code. In these systems, users may additionally undergo multiple cycles of rewant and remix as they iterate on what can be accomplished with the new tools and decide on preferred experimental styles to remix. Conversely, they may just be content with only applying familiar methods and mental models, and are unwilling to reconsider traditional strategies. Some users may also not want to share their work as examples. The collection of inspirational examples may thus become saturated in one type of aesthetic or become an onus of the system. We also note that the user model is only applicable when a reported learning curve is present in a creative tool, and in most instances, systems are designed to mitigate learning curves. However, in instances

when learning is inevitable for the user, we hope that the model can provide useful insights and help direct purposeful design choices that augment the artist's drawing experience.

7 LIMITATIONS AND FUTURE WORK

Since participants only used the tool for 80 minutes, a third of which was spent familiarizing themselves with the application and learning about filters, future longitudinal studies with a smaller focus group can better reveal more nuances about artist thought processes, creative possibilities enabled by this type of tool, and how the proposed user model evolves. Our system is also limited by the simplicity of its canvas interface. Participants felt that the lack of layers and limited variety of drawing tools like shapes or lasso prevented them from creating more complex illustrations. Perhaps the filter editor could instead be restructured as a plugin to an existing more comprehensive drawing tool.

Current image-sharing platforms rarely support the full extent of vector-based media. While a standalone animated vector file can be opened and loaded in modern web browsers such as Chrome and Safari, it cannot be uploaded and previewed on many portfolio sites. One participant (P8) commented on this, "I can't send animated SVGs to people and post it on Instagram; I'd have to screen-record in order to show people what I made," essentially converting the file format back to raster and removing edit capabilities. This lack of widespread systematic support deters both system designers and artists, as artwork created using filtered.ink cannot be posted on sites artists would typically share their work on.

8 CONCLUSION

filtered.ink attempts to expand the boundaries of vector illustration by empowering artists to design lightweight textures and animations that can be applied to brush strokes. These new properties enable vector illustrations to depart from standard minimalist, logo-like, and flat styles to instead portray rich moods and atmosphere through the medley of lighting, material, and dynamic effects. To achieve this, filtered.ink relies on a live node graph-based editor that decomposes filter code of a type of vector graphic, SVGs, into semantic components. This interface lets users bypass coding prerequisites to manipulate their illustration instrument so that it aligns with their visual mental model. Using stateful looping through filter parameter values, our system also allows illustrators without animation backgrounds to create repeating animations in vector. We collect observations from a task-based usability evaluation of filtered.ink with hobbyist and professional artists into a user model on participant workflow patterns and stylistic outcomes for initial usage of creative systems with technical learning curves. Users in this model are able to create personally satisfactory art by evolving familiar mental models and workflows into new ones. By creating, recombining, and sharing painterly, dynamic vector illustrations, users are afforded a new way to express themselves on the web.

ACKNOWLEDGMENTS

This work is supported by the National Science Foundation under Grant No. IIS-1552663. We thank Ji Won Chung, Zainab Iftikhar, Talie Massachi, Shaun Wallace, James Tompkin, and Daniel Ritchie for their feedback and intellectual support.

REFERENCES

[1] Vladimir Agafonkin. 2020. Simplify.js. https://github.com/mourner/simplify-js.
[2] Jorge Aznar. 2016. FILDROP. http://jorgeatgu.github.io/svg-filters/.
[3] Alexandre Carlier, Martin Danelljan, Alexandre Alahi, and Radu Timofte. 2020. DeepSVG: A Hierarchical Generative Network for Vector Graphics Animation. arXiv:2007.11301 [cs.CV]
[4] Alessandra Chiarini. 2016. The Multiplicity of the Loop: The Dialectics of Stillness and Movement in the Cinemagraph. *Comunicazioni sociali* 38, 1 (2016), 87–92. https://doi.org/10.1400/240293
[5] Ayan Das, Yongxin Yang, Timothy M. Hospedales, Tao Xiang, and Yi-Zhe Song. 2021. Cloud2Curve: Generation and Vectorization of Parametric Sketches. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, Nashville, TN, USA, 7088–7097. https://doi.org/10.1109/CVPR46437.2021.00701
[6] Julie Dorsey, Songhua Xu, Gabe Smedresman, Holly Rushmeier, and Leonard McMillan. 2007. The Mental Canvas: A Tool for Conceptual Architectural Design and Analysis. In *15th Pacific Conference on Computer Graphics and Applications (PG'07)*. IEEE, Maui, Hawaii, 201–210. https://doi.org/10.1109/PG.2007.64
[7] Jakub Fišer, Michal Lukáč, Ondřej Jamriška, Martin Čadík, Yotam Gingold, Paul Asente, and Daniel Sýkora. 2014. Color Me Noisy: Example-based Rendering of Hand-colored Animations with Temporal Noise Control. *Computers Graphics Forum (EGSR 2014)* 33, 4 (2014), 1–10. https://doi.org/10.1111/cgf.12407
[8] Christoph M. Flath, Sascha Friesike, Marco Wirth, and Frederic Thiesse. 2017. Copy, Transform, Combine: Exploring the Remix as a Form of Innovation. *Journal of Information Technology* 32, 4 (2017), 306–325. https://doi.org/10.1057/s41265-017-0043-9
[9] Neil Fraser. 2015. Ten Things We've Learned from Blockly. In *Proceedings of the 2015 IEEE Blocks and Beyond Workshop*. IEEE Computer Society, USA, 49–50. https://doi.org/10.1109/BLOCKS.2015.7369000
[10] Brian Hempel and Ravi Chugh. 2016. Semi-Automated SVG Programming via Direct Manipulation. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology* (Tokyo, Japan) *(UIST '16)*. Association for Computing Machinery, New York, NY, USA, 379–390. https://doi.org/10.1145/2984511.2984575
[11] Brian Hempel, Justin Lubin, and Ravi Chugh. 2019. Sketch-n-Sketch: Output-Directed Programming for SVG. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology* (New Orleans, LA, USA) *(UIST '19)*. Association for Computing Machinery, New York, NY, USA, 281–292. https://doi.org/10.1145/3332165.3347925
[12] Scarlett R. Herring, Chia-Chen Chang, Jesse Krantzler, and Brian P. Bailey. 2009. Getting Inspired! Understanding How and Why Examples Are Used in Creative Design Practice. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Boston, MA, USA) *(CHI '09)*. Association for Computing Machinery, New York, NY, USA, 87–96. https://doi.org/10.1145/1518701.1518717
[13] Yiwei Hu, Julie Dorsey, and Holly Rushmeier. 2019. A Novel Framework for Inverse Procedural Texture Modeling. *ACM Trans. Graph.* 38, 6, Article 186 (nov 2019), 14 pages. https://doi.org/10.1145/3355089.3356516
[14] Yiwei Hu, Chengan He, Valentin Deschaintre, Julie Dorsey, and Holly Rushmeier. 2022. An Inverse Procedural Modeling Pipeline for SVBRDF Maps. *ACM Trans. Graph.* 41, 2, Article 18 (jan 2022), 17 pages. https://doi.org/10.1145/3502431
[15] Adobe Inc. 2018. After Effects. https://www.adobe.com/products/aftereffects.html
[16] Adobe Inc. 2019. Illustrator. https://adobe.com/products/illustrator
[17] Client IO. 2019. SVG Filter Builder. https://svgfilters.com/.
[18] Jennifer Jacobs, Joel Brandt, Radomír Mech, and Mitchel Resnick. 2018. Extending manual drawing practices with artist-centric programming tools. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3173574.3174164
[19] Jennifer Jacobs, Sumit Gogia, Radomír Mundefinedch, and Joel R. Brandt. 2017. Supporting Expressive Procedural Art Creation through Direct Manipulation. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (Denver, Colorado, USA) *(CHI '17)*. Association for Computing Machinery, New York, NY, USA, 6330–6341. https://doi.org/10.1145/3025453.3025927
[20] Neel Joshi, Sisil Mehta, Steven Drucker, Eric Stollnitz, Hugues Hoppe, Matt Uyttendaele, and Michael Cohen. 2012. Cliplets: Juxtaposing Still and Dynamic Imagery. In *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology* (Cambridge, Massachusetts, USA) *(UIST '12)*. Association for Computing Machinery, New York, NY, USA, 251–260. https://doi.org/10.1145/2380116.2380149
[21] Hyeonsu B. Kang, Gabriel Amoako, Neil Sengupta, and Steven P. Dow. 2018. Paragon: An Online Gallery for Enhancing Design Feedback with Visual Examples. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (Montreal QC, Canada) *(CHI '18)*. Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3173574.3174180
[22] Rubaiat Habib Kazi, Fanny Chevalier, Tovi Grossman, and George Fitzmaurice. 2014. Kitty: sketching dynamic and interactive illustrations. In *Proceedings of the 27th annual ACM symposium on User interface software and technology*.

Association for Computing Machinery, New York, NY, USA, 395–405. https://doi.org/10.1145/2642918.2647375

[23] Rubaiat Habib Kazi, Fanny Chevalier, Tovi Grossman, Shengdong Zhao, and George Fitzmaurice. 2014. Draco: Bringing Life to Illustrations with Kinetic Textures. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Toronto, Ontario, Canada) *(CHI '14)*. Association for Computing Machinery, New York, NY, USA, 351–360. https://doi.org/10.1145/2556288.2556987

[24] Rubaiat Habib Kazi, Tovi Grossman, Nobuyuki Umetani, and George Fitzmaurice. 2016. SKUID: Sketching Dynamic Drawings Using the Principles of 2D Animation. In *Proceedings of the 4th International Conference on Mobile and Ubiquitous Multimedia* (Christchurch, New Zealand) *(MUM '05)*. Association for Computing Machinery, New York, NY, USA, 69–77. https://doi.org/10.1145/2897839.2927410

[25] Rubaiat Habib Kazi, Takeo Igarashi, Shengdong Zhao, and Richard Davis. 2012. Vignette: Interactive Texture Design and Manipulation with Freeform Gestures for Pen-and-Ink Illustration. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Austin, Texas, USA) *(CHI '12)*. Association for Computing Machinery, New York, NY, USA, 1727–1736. https://doi.org/10.1145/2207676.2208302

[26] Michele Knobel and Colin Lankshear. 2008. Remix: The Art and Craft of Endless Hybridization. *Journal of Adolescent & Adult Literacy* 52, 1 (2008), 22–33. http://www.jstor.org/stable/30139647

[27] Jan-Peter Krämer, Michael Hennings, Joel Brandt, and Jan Borchers. 2016. An Empirical Study of Programming Paradigms for Animation. In *Proceedings of the 9th International Workshop on Cooperative and Human Aspects of Software Engineering* (Austin, Texas) *(CHASE '16)*. Association for Computing Machinery, New York, NY, USA, 58–61. https://doi.org/10.1145/2897586.2897597

[28] Chinmay Kulkarni, Steven P. Dow, and Scott R. Klemmer. 2014. Early and Repeated Exposure to Examples Improves Creative Work. In *Design Thinking Research: Building Innovation Eco-Systems*, Larry Leifer, Hasso Plattner, and Christoph Meinel (Eds.). Springer International Publishing, Cham, 49–62. https://doi.org/10.1007/978-3-319-01303-9_4

[29] Sang Won Lee, Yujin Zhang, Isabelle Wong, Yiwei Yang, Stephanie D O'Keefe, and Walter S Lasecki. 2017. SketchExpress: Remixing Animations for More Effective Crowd-Powered Prototyping of Interactive Interfaces. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*. Association for Computing Machinery, New York, NY, USA, 817–828. https://doi.org/10.1145/3126594.3126595

[30] Jingyi Li, Joel Brandt, Radomír Mech, Maneesh Agrawala, and Jennifer Jacobs. 2020. Supporting Visual Artists in Programming through Direct Inspection and Control of Program Execution. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3313831.3376765

[31] Jingyi Li, Sonia Hashim, and Jennifer Jacobs. 2021. What We Can Learn From Visual Artists About Software Development. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, USA, 1–14. https://doi.org/10.1145/3411764.3445682

[32] Yingjia Li, Xiao Zhai, Fei Hou, Yawen Liu, Aimin Hao, and Hong Qin. 2019. Vectorized painting with temporal diffusion curves. *IEEE Transactions on Visualization and Computer Graphics* 27, 1 (2019), 228–240. https://doi.org/10.1109/TVCG.2019.2929808

[33] Raphael Gontijo Lopes, David Ha, Douglas Eck, and Jonathon Shlens. 2019. A Learned Representation for Scalable Vector Graphics. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, Seoul, South Korea, 7930–7939. https://doi.org/10.1109/ICCV.2019.00802

[34] Damien Masson, Jo Vermeulen, George Fitzmaurice, and Justin Matejka. 2022. Supercharging Trial-and-Error for Learning Complex Software Applications. In *CHI Conference on Human Factors in Computing Systems* (New Orleans, LA, USA) *(CHI '22)*. Association for Computing Machinery, New York, NY, USA, Article 381, 13 pages. https://doi.org/10.1145/3491102.3501895

[35] Mozilla. 2021. <filter>. https://developer.mozilla.org/en-US/docs/Web/SVG/Element/filter.

[36] Lora Oehlberg, Wesley Willett, and Wendy E. Mackay. 2015. Patterns of Physical Design Remixing in Online Maker Communities. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (Seoul, Republic of Korea) *(CHI '15)*. Association for Computing Machinery, New York, NY, USA, 639–648. https://doi.org/10.1145/2702123.2702175

[37] Amal Dev Parakkat, Marie-Paule R Cani, and Karan Singh. 2021. Color by numbers: Interactive structuring and vectorization of sketch imagery. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, USA, 1–11. https://doi.org/10.1145/3411764.3445215

[38] The Inkscape Project. 2020. Inkscape. https://inkscape.org

[39] Jing Qian, Tongyu Zhou, Meredith Young-Ng, Jiaju Ma, Angel Cheung, Xiangyu Li, Ian Gonsher, and Jeff Huang. 2021. Portalware: Exploring Free-Hand AR Drawing with a Dual-Display Smartphone-Wearable Paradigm. In *Designing Interactive Systems Conference 2021* (Virtual Event, USA) *(DIS '21)*. Association for Computing Machinery, New York, NY, USA, 205–219. https://doi.org/10.1145/3461778.3462098

[40] Pradyumna Reddy, Michael Gharbi, Michal Lukac, and Niloy J. Mitra. 2021. Im2Vec: Synthesizing Vector Graphics without Vector Supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, Los Alamitos, CA, USA, 7342–7351. https://doi.org/10.1109/CVPR46437.2021.00726

[41] Mitchel Resnick, John Maloney, Andrés Monroy-Hernández, Natalie Rusk, Evelyn Eastmond, Karen Brennan, Amon Millner, Eric Rosenbaum, Jay Silver, Brian Silverman, and Yasmin Kafai. 2009. Scratch: Programming for All. *Commun. ACM* 52, 11 (nov 2009), 60–67. https://doi.org/10.1145/1592761.1592779

[42] John Rieman. 1996. A Field Study of Exploratory Learning Strategies. *ACM Trans. Comput.-Hum. Interact.* 3, 3 (September 1996), 189–218. https://doi.org/10.1145/234526.234527

[43] Vidya Setlur, Yingqing Xu, Xuejin Chen, and Bruce Gooch. 2005. Retargeting vector animation for small displays. In *Proceedings of the 4th international conference on mobile and ubiquitous multimedia*. Association for Computing Machinery, New York, NY, USA, 69–77. https://doi.org/10.1145/1149488.1149500

[44] Liang Shi, Beichen Li, Miloš Hašan, Kalyan Sunkavalli, Tamy Boubekeur, Radomir Mech, and Wojciech Matusik. 2020. Match: Differentiable Material Graphs for Procedural Material Capture. *ACM Trans. Graph.* 39, 6, Article 196 (nov 2020), 15 pages. https://doi.org/10.1145/3414685.3417781

[45] Yang Shi, Zhaorui Li, Lingfei Xu, and Nan Cao. 2021. Understanding the Design Space for Animated Narratives Applied to Illustrations. In *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, USA, 1–6. https://doi.org/10.1145/3411763.3451840

[46] Ben Shneiderman. 2007. Creativity Support Tools: Accelerating Discovery and Innovation. *Commun. ACM* 50, 12 (dec 2007), 20–32. https://doi.org/10.1145/1323688.1323689

[47] Pao Siangliulue, Joel Chan, Krzysztof Z. Gajos, and Steven P. Dow. 2015. Providing Timely Examples Improves the Quantity and Quality of Generated Ideas. In *Proceedings of the 2015 ACM SIGCHI Conference on Creativity and Cognition* (Glasgow, United Kingdom) *(C&C '15)*. Association for Computing Machinery, New York, NY, USA, 83–92. https://doi.org/10.1145/2757226.2757230

[48] Ivan E. Sutherland. 1964. Sketch Pad a Man-Machine Graphical Communication System. In *Proceedings of the SHARE Design Automation Workshop (DAC '64)*. Association for Computing Machinery, New York, NY, USA, 6.329–6.346. https://doi.org/10.1145/800265.810742

[49] SVGator Team. 2022. SVGator. https://www.svgator.com/

[50] John Thompson, Zhicheng Liu, Wilmot Li, and John Stasko. 2020. Understanding the design space and authoring paradigms for animated data graphics. In *Computer Graphics Forum*, Vol. 39. Wiley Online Library, Norrköping, Sweden, 207–218. https://doi.org/10.1111/cgf.13974

[51] Kathleen Tuite and Adam Smith. 2021. Emergent Remix Culture in an Anonymous Collaborative Art System. *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment* 8, 5 (June 2021), 16–23. https://ojs.aaai.org/index.php/AIIDE/article/view/12572

[52] Tom Van Laerhoven, Fabian Di Fiore, William Van Haevre, and Frank Van Reeth. 2011. Paint-on-glass animation: the fellowship of digital paint and artisanal control. *Computer Animation and Virtual Worlds* 22, 2-3 (2011), 325–332. https://doi.org/10.1002/cav.406

[53] Shaun Wallace, Brendan Le, Luis A. Leiva, Aman Haq, Ari Kintisch, Gabrielle Bufrem, Linda Chang, and Jeff Huang. 2020. Sketchy: Drawing Inspiration from the Crowd. *Proc. ACM Hum.-Comput. Interact.* 4, CSCW2, Article 172 (oct 2020), 27 pages. https://doi.org/10.1145/3415243

[54] David Weintrop and Uri Wilensky. 2017. Comparing Block-Based and Text-Based Programming in High School Computer Science Classrooms. *ACM Trans. Comput. Educ.* 18, 1, Article 3 (oct 2017), 25 pages. https://doi.org/10.1145/3089799

[55] Haijun Xia, Bruno Araujo, Tovi Grossman, and Daniel Wigdor. 2016. Object-Oriented Drawing. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems* (San Jose, California, USA) *(CHI '16)*. Association for Computing Machinery, New York, NY, USA, 4610–4621. https://doi.org/10.1145/2858036.2858075

[56] Lixiu Yu and Jeffrey V. Nickerson. 2011. Cooks or Cobblers? Crowd Creativity through Combination. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Vancouver, BC, Canada) *(CHI '11)*. Association for Computing Machinery, New York, NY, USA, 1393–1402. https://doi.org/10.1145/1978942.1979147

[57] Zhengxia Zou, Tianyang Shi, Shuang Qiu, Yi Yuan, and Zhenwei Shi. 2021. Stylized Neural Painting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, Nashville, TN, USA, 15689–15698. https://doi.org/10.1109/CVPR46437.2021.01543