Eurographics Conference on Visualization (EuroVis) 2020
M. Gleicher, T. Landesberger von Antburg, and I. Viola
(Guest Editors)

Volume 39 (2020), Number 3

Understanding the Design Space and Authoring Paradigms for Animated Data Graphics

J. Thompson[1], Z. Liu[2], W. Li[2] and J. Stasko[1]

[1]Georgia Institute of Technology, Atlanta, Georgia, USA
[2]Adobe Research, Seattle, Washington, USA

Abstract

Creating expressive animated data graphics often requires designers to possess highly specialized programming skills. Alternatively, the use of direct manipulation tools is popular among animation designers, but these tools have limited support for generating graphics driven by data. Our goal is to inform the design of next-generation animated data graphic authoring tools. To understand the composition of animated data graphics, we survey real-world examples and contribute a description of the design space. We characterize animated transitions based on object, graphic, data, and timing dimensions. We synthesize the primitives from the object, graphic, and data dimensions as a set of 10 transition types, and describe how timing primitives compose broader pacing techniques. We then conduct an ideation study that uncovers how people approach animation creation with three authoring paradigms: keyframe animation, procedural animation, and presets & templates. Our analysis shows that designers have an overall preference for keyframe animation. However, we find evidence that an authoring tool should combine these three paradigms as designers' preferences depend on the characteristics of the animated transition design and the authoring task. Based on these findings, we contribute guidelines and design considerations for developing future animated data graphic authoring tools.

CCS Concepts
• ***Human-centered computing*** $\rightarrow$ *Visualization theory, concepts and paradigms; Empirical studies in visualization;*

1. Introduction

Animated graphics are a major type of animation that focuses on the motion of graphic design elements. The graphics can be bitmap images, but vector graphics in the form of geometric shapes and text are more prevalent. These vector-based graphical objects can represent abstract concepts (e.g., a company name) or real-life objects and characters. To create an animation, designers specify how the properties and behaviors of the graphical objects change over time. This type of animation has many purposes such as to transition between states of an interface, to highlight changes in a view, to attract viewers' attention, and to facilitate storytelling. Examples of such animation are numerous and range from logo animation to particle system simulation to animated illustration.

A variety of commercial and research tools have been developed for creating 2D and 3D motion graphics [Ado20a], animation in user interface prototyping [HS93, Inv20, Pri20, Tum20, Ado20c], animation in presentation [App20, Mic20], and animated illustration and diagrams [DCL08, KCG*14, KCGF14, ZS03]. In these tools, the animated behavior is usually defined by designers drawing upon their artistic expertise. Prevalent authoring paradigms for animated graphics include:

- keyframe animation: specify properties of graphical objects at certain points of time by setting a set of keyframes, frames in between two keyframes are generated by tweening.
- procedural animation: generate animation of large number of animated objects with a set of behavior parameters.
- presets and templates: apply predefined animation effects and configurations to objects.

Our focus in this work is on a specific type of animated graphics known as *animated data graphics*. In this area, an underlying structured data set is being visualized and animation is part of the presentation. Much like an information visualization depicts data through graphical objects, in animated data graphics a visualization changes and updates based, at least in part, on the values within an underlying data set.

Animation may be necessary or at least desirable to be used for a variety of reasons. First, the data itself may be changing. Alternately, time may be a fundamental attribute of the data set [AMM*08] and animation is used to represent temporal changes. Animation additionally can enhance a data story by encoding data (reveal relationships, encode emotions and data attributes), supporting discourse (narrative flow, highlight content), and improving user experience (keep users engaged, provide visual comfort and aesthetics, hook the user) [CRP*16]. Animated data graphics are popular in digital journalism for narrative purposes. Researchers and designers have also used them to explain complex concepts and problems (e.g., algorithm animation [Bro88,KS02]). Custom analytics tools also employ animated transitions to update dashboards [WSC17,Few06].

Compared to general animated graphics, authoring tools for animated data graphics are still relatively scarce. To create sophisticated animated designs, developers often employ programming libraries [BOH11,RF06]. Some tools try to combine graphical interfaces with scripting for creating animated data graphics [Ado19], but the range of expression is limited. Non-programming tools [Ado20a, Kil19] provide predefined templates, which constrain the variety of animation designs.

What additional complexities are introduced to animation when data becomes an integral part of animation design? Moreover, what can we learn from current animation tools to inform the design of next generation authoring tools for animated data graphics? How do the characteristics of animated data graphics influence designers' preference of authoring approaches? To help answer these questions, we analyze 52 examples of animated data graphics collected in the wild. The analysis of these real-world examples yields a design space, expressed as primitives in four dimensions: *object*, *graphics*, *data*, and *timing*. These primitives provide building blocks for higher-level compositions such as transition types and pacing techniques. Under the backdrop of the animation authoring paradigms and the design space, we conduct an ideation study to understand how designers conceptually approach authoring animated data graphics. The study results show that keyframe animation is a familiar authoring paradigm preferred by many participants in a majority of tasks. However, we find evidence that an authoring tool should combine paradigms as the other two paradigms are indispensable for certain tasks.

2. Background and Related Work

2.1. Authoring Paradigms for Animated Graphics

There are three dominant animation authoring paradigms that are relevant for animated data graphics: keyframe animation, procedural animation, and presets and templates. Performance animation, where users directly act out the desired motion, is another common authoring paradigm. However, it is typically applied for animating characters rather than large collections of graphical elements, which are common in data graphics. An authoring paradigm can be implemented in different input modalities and interaction techniques (e.g., natural language interfaces, programming languages, Graphical User Interfaces (GUI), sketching, gesturing). Here, we review related research and existing commercial systems that adopt keyframing, procedural animation, and presets and templates.

2.1.1. Keyframe Animation

Motion graphics tools like After Effects [Ado20a] allow users to define properties (e.g., position, scale, opacity) of graphical objects at specific points in time in the form of a keyframe. Intermediate frames are created by interpolating/tweening the visual properties between defined keyframes. The graphical user interface for keyframing is usually in the form of a timeline (Figure 1a) coupled with a property inspector. In User Interface (UI) prototyping tools, it is possible to

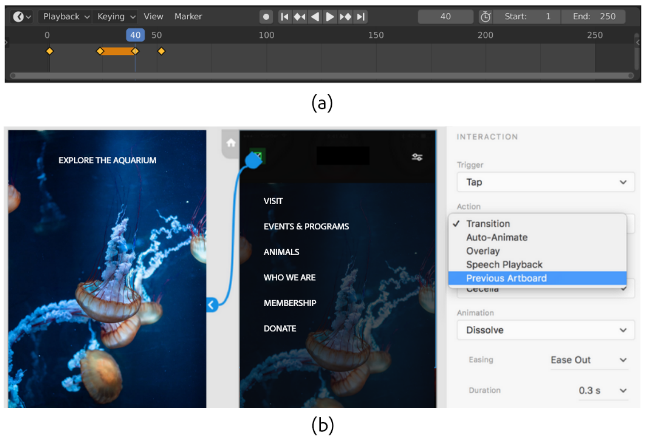

Figure 1: *(a) the Timeline in Blender [Ble02] where users can define keyframes for the active object. The diamond shapes represent keyframes; (b) Users create "auto-animate" transition effects between two artboards in Adobe XD [Ado20c] by drawing a blue connector.*

treat each UI state as a keyframe. Instead of defining behavior explicitly on a timeline, sometimes simpler interfaces are used (e.g. users draw a connector to link two artboards representing two keyframes) (Figure 1b). Programming toolkits such as Rekapi [Kah17] support the creation of keyframe animation through coding.

2.1.2. Procedural Animation

While keyframing gives designers precise and direct control over the properties and behaviors of graphical objects, it can be tedious to specify coordinated motions of a large number of objects. Procedural animation is a popular paradigm for animations involving many objects, such as particle systems (e.g. fire, rainfall), flocking (e.g. a school of fish) and stochastic motion (e.g. leaves in the wind) [KCG*14]. These animating behaviors are typically created procedurally: starting with an initial condition, an engine (often called an "emitter" or "oscillator") generates motion by adjusting a set of parameters (e.g. wind strength, object's stiffness). The algorithmic outcome may be unpredictable. User interfaces for procedural animation come in many forms: programming languages (e.g. tweening in D^3), WIMP (Windows, Icons, Menus, Pointer) components (Figure 2a), sketching (Figure 2b), and node-based visual programming (e.g. TouchDesigner [Tou20]).

2.1.3. Presets and Templates

For novice and casual users, it is not always beneficial to use a powerful tool that forces them to start from scratch. Reusable animation presets and templates can significantly lower the learning threshold and reduce the time and effort needed, and these predefined animation types are sufficient in many use cases. Presentation tools offer animation effects both at the frame level (e.g. slide transition effects) and at the object level (e.g. fly-in or bounce) (Figure 3). Parameters of these effects are also exposed for user manipulation. Similarly, motion graphics tools provide such functionality (e.g. text animating presets and motion graphics templates in After Effects [Ado20a]).

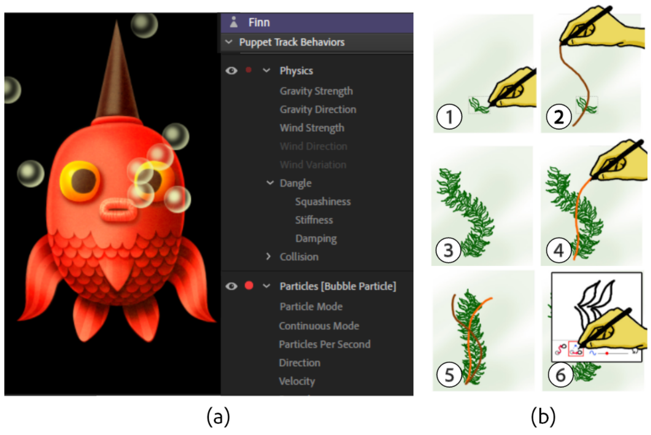

Figure 2: *(a) interface in Adobe Character Animator [Ado20b] for users to specify parameters of particle animation (bubbles from the fish mouth) and physical properties of the fish body; (b) steps to create oscillating motions of leaves in Draco [KCG*14].*

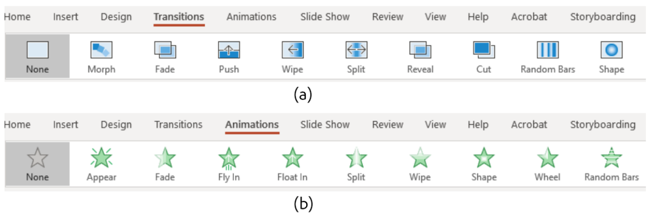
Home Insert Design Transitions Animations Slide Show Review View Help Acrobat Storyboarding
None Morph Fade Push Wipe Split Reveal Cut Random Bars Shape
(a)
Home Insert Design Transitions Animations Slide Show Review View Help Acrobat Storyboarding
None Appear Fade Fly In Float In Split Wipe Shape Wheel Random Bars
(b)

Figure 3: *slide transition effects and object animation presets in Microsoft Powerpoint [Mic20]*

Users simply choose from a list of available presets and apply to their own graphical objects.

It is common for an animation tool to support multiple authoring paradigms. For example, After Effects [Ado20a] supports both keyframing and presets/templates. Complex animations may also be composed by combining different paradigms. For example, one may need to add procedurally generated particle motion on top of a keyframe animation.

2.2. Tools for Animated Data Graphics

Tools for creating animated data graphics are sparse. Visualization designers often rely on programming toolkits [BOH11, RF06] or custom scripts to create sophisticated data-driven animations. For example, D^3 supports the specification of data-driven transitions based on the "enter, update, exit" paradigm. D^3 also supports a number of interpolation functions that animate visual properties for a declared function of duration, delay, and easing. Building on top of ggplot2 [Wic09], gganimate offers programmatic ways to describe animation [LPR19]. Progamming toolkits such as these are very expressive, but coding has a steep learning curve and demands significant time and cognitive effort.

Non-programming tools tend to rely on templates for the creation of animated data graphics. DataClips [ARL*17] allows authors to create animated data videos by composing a sequence of predefined animation and visualization combinations. After Effects [Ado20a] has data-driven motion graphics templates. Flourish [Kil19] has recently introduced animated transitions to "Talkies" [Cla19], allowing for audio-driven visualizations that interpolate marks' properties between narrative points. However, these tools only support a limited set of visualization templates; DataClips and Flourish do not allow for the design of pacing or animation effects. The motivation of this research is to understand how we can go beyond the template paradigm and support the creation of more nuanced and expressive animated data graphics.

2.3. Taxonomies on Animated Data Graphics

Many taxonomies exist to understand the different types of changes in animated data graphics. Heer and Robertson defined a taxonomy of animated transitions in statistical data graphics [HR07]. Their taxonomy defines 7 types of animated transitions by considering the syntactic or semantic operators one might apply to a visualization. Fisher [Fis10] adapted this taxonomy and proposed a list of six animation types in visualization. DataClips [AHRL*15] identified high-level building blocks of data videos expressed in visualization type $\times$ animation type combinations. Chalbi [Cha18] distinguished between data-driven changes and visual-driven changes, and enumerated animated changes at the level of low-level components. Chevalier et al. [CRP*16] went beyond data graphics to examine the different roles played in animation in user interfaces, so that novel uses of animation and research opportunities could be identified. The primary goal of our analysis of animated data graphics examples is not to build a comprehensive taxonomy. Instead, we aim to identify the primitives to be used in the authoring of animated data graphics along four dimensions: object, graphics, data and timing. Exemplar compositions of these primitives, such as transition types and pacing techniques, are useful higher-level constructs that can be included in authoring tools.

2.4. Studies on Authoring Paradigms and Workflows

Amini et al. observed how designers create data storyboards based on data facts [AHRL*15]. They focused on the activities and processes involved in the creation of storyboards. Kramer et al. studied how programming paradigms for animation affect developers' productivity [KHBB16]. They compared two paradigms: declarative (which corresponds to keyframing) and procedural (which is usually implemented as requestAnimationFrame in Javascript), and found that developers could implement animations faster using a declarative language, but declarative languages could lead to unexpected animation behavior. In our study, we do not wish for the participants to be restricted to a specific input modality or user interface. Instead, we focus on authoring paradigms at a conceptual level. Our ideation study is thus more open-ended, and our methodology resembles more of that used in the Data Illustrator project [LTW*18], where designers were asked to demonstrate workflows using tools of their choice. Since data sketching offers an unrestricted way to solicit user ideas [RHR16, WHC15], we also encouraged our users to sketch their ideas on paper whenever possible.

3. Survey of Animated Data Graphics

To begin, we will introduce terminology to be used throughout the remainder of the article. Animated data graphics are composed of *animated transitions* on data graphic elements. An *animated transition* is the interpolated change between two *visual states*. We further breakdown these terms as follows:

- A *visual state* can be thought of as a data graphic at rest. Visual states include *all* information about the data graphic's static design.
- A *transition* is the uninterrupted change from one visual state of a data graphic to another.
- *Animation* is the sequence of intermediary states between a transition that is typically perceived as continuous movement.

To understand the design space of animated data graphics we surveyed examples from online sources. The purpose of our survey is to *(i)* identify composable primitives of animated data graphics, and *(ii)* recognize common compositions of those primitives. Our survey includes data narrative articles, data videos, visualization slideshows, and interactive maps. We collected an initial set of 78 examples from the following sources: the Kantar Information is Beautiful Awards [Kan20]; media outlets including the New York Times [New18], the Pudding [Pud20], the Google News Lab [Goo20]; and well-known freelance designers such as Nadieh Bremer [Bre20], Neil Halloran [Hal20], Shirley Wu [Wu20], and Moritz Stefaner [Ste20].

We exclude examples where all animated transitions are unrelated to data. For example, we exclude user interface transitions such as highlighting buttons or expanding menus. We also exclude animated transitions that only occur at the frame level – meaning none of the constituent objects transition. By only including examples with data-related animated transitions, we refined our initial pool to the final collection of 52 examples. This survey is not exhaustive, however our goal is not to catalogue all animated data graphics but to inform the design of authoring tools. To that end, our survey includes a diverse set of data graphic forms and animated transition designs.

To analyze these examples, we first identified their unique animated transition instances. We identified instances based on the following criteria: *(i)* an object undergoes an animated transition, *(ii)* the animated transition is data-related, *(iii)* repetitive animated transitions count as a single instance (e.g., circles in a scatterplot change y-position). When analyzing these animation instances, we set out to identify building blocks for future authoring tools that go beyond templates. Related taxonomies [HR07, ARL*17] provide transition types that are well-suited for animation templates. While templates are easy to use and understand, they are difficult to customize. We seek to identify composable primitives for future authoring tools that are generalizable across datasets and visualization forms.

During weekly meetings over a period of three months, we iterated on these primitives until we could accurately describe each animated transition instance. During this iterative process we sought to balance the granularity of our primitives. Too high-level and the primitives would resemble animation templates; while low-level primitives would not be generalizable across datasets and visualization forms. We identified composable primitives from each of four dimensions: *object*, *graphic*, *data*, and *timing*. The *object* dimension refers to *what* type of graphic object transitions during the animated transition. The *graphic* and *data* dimensions describe *how* the object transitions from one *visual state* to the next. Finally, the *timing* dimension consists of relative timing concepts to compose animation designs and pace sequences of animated transitions.

4. Design Space of Animated Data Graphics

Here we provide a description of the design space for animated data graphics. We characterize the design space by identifying composable primitives across four dimensions: *object*, *graphic*, *data* and *timing*. The primitives combine to form animated transition designs. We provide a list of exemplar compositions in the form of transition types (Section 4.2.1) and pacing techniques (Section 4.2.2) that we identified during our survey of examples.

4.1. Primitives

4.1.1. Dimension 1: Object Type

The *object* dimension describes *what* type of graphic object undergoes a transition. We propose the following set of 5 object types, delineated by their *role* in a data graphic and their unique properties. The same graphical element can be used in a variety of ways, differing by its role, relation to data, constitutent parts and attributes. For example, a rectangle can be used as a glyph to represent data and encode data values using its visual attributes; a rectangle may also be used as a legend or an annotation.

Glyphs are graphical marks (e.g., lines, rectangles, text, images, groups) representing one or more data tuples. A glyph's visual appearance may encode data values.

Groups are collections of glyphs. Glyphs within a group are often arranged in a spatial layout.

Axes & Legends are the visual representation of scales that map data values to visual properties. They explain how data maps to visual space of the data graphic.

Annotations are auxiliary elements that help explain key insights and contextual information about the data graphic to the audience. Annotations contain text, shape, and image elements and target different components of the data graphic (e.g., specific glyph, series of glyphs, visual substrate, entire graphic). Annotations include labels, captions, tooltips, and footnotes. Annotations are not bound to data, although they may depend on data attribute-values from a target glyph.

Cameras provide a configurable vantage point of the data graphic. The camera (or viewport) projects the scene graph onto a view based on the camera's configuration. The projection attributes depends on camera type, but can include: focal point, field of view, zoom level, and rotation. Changes to these parameters result in pan, zoom, and rotation actions.

4.1.2. Dimension 2: Graphics

The *graphics* dimension describes *how* an object changes from one visual state to the next. It is concerned with the constituent visual elements, their visual attributes, and configurations that compose a data graphic scene. Initially, we considered describing the low-level visual properties that change for each object (e.g., visibility, position, opacity, fill color, text content). However, this approach only provides a taxonomy of the visual attributes that transition in our set of examples. Instead, we provide 3 primitives that describe how an object visually changes.

Visual Presence is the existence of an object. When an object is added or becomes visible, we say it "enters" the scene graph. Conversely when an object is removed, we say it "exits."

Visual Attributes are the visual channels of objects such as position, fill color, stroke, opacity, etc. that are set by the designer and unrelated to data encodings. Unless overridden by data encodings or configurations, glyph instances share the same visual attribute values. Visual attributes differ for graphical elements (e.g., text elements have different visual attributes than line elements).

Configurations are the parameters of an object that are not directly visible in objects. Configurations differ from visual attributes because the parameter value is not directly expressed as a visual value. For example, a group can have a layout configuration, specifying how its constituent glyphs should be positioned.

4.1.3. Dimension 3: Data

The *data* dimension describes *how* the underlying data changes in a data graphic. Modifications to data mappings can change visual properties of objects. For example, if the bound data attribute is changed in a mapping, the glyphs may express a new visual value for the newly bound data values. Change in the underlying data at times results in visual changes to glyphs, however this differs from the *graphic* dimension where each glyph expresses the same visual attributes regardless of data.

Data Presence is the existence of data tuples. Data is added or removed to a data graphic through manifestations in objects. For example, glyph instances are bound to one or more data tuples.

Data Encodings are the functions that transform data attribute values into visual property values for each glyph instance. Data encodings can be shared across groups of glyphs.

Data Transforms are the operators that manipulate datasets into new forms to then be attached to graphical objects. Data transforms include data nesting, aggregation, and linking. These operations allow for the specification of data graphic designs beyond the raw dataset.

Data Targets are the data focal points for other objects. For example, annotations such as labels and tooltips target a glyph for a specific data tuple.

Data Queries are predicates applied to a dataset that generate inclusion and exclusion selections. The design specifies how the corresponding glyphs for these two selections will be visually altered. For example, filtering temporally removes the exclusion selection from the scene graph, while highlighting visually emphasizes the inclusion glyphs and/or de-emphasizes the exclusion glyphs.

4.1.4. Dimension 4: Timing

The *timing* of an animation describes the pace at which visual properties successively move from start to end of a transition. Although a transition has the same start and end visual states, animation provides diverse opportunities to illustrate between those states. We describe timing of animations based on 4 primitives. The primitives rely on a relative notion of timing. For example, the *duration* of an animation is defined as the relative amount of time that transpires between the start and end of an animation. Relative timing allow designers to compose successive animations into larger animation compositions.

Triggers are events that initiate an animation. Triggers provide an initial reference point. Triggers include the following: a specific timestamp indicating the start or end of another animation, repetitive temporal events such as a clock, or navigation inputs such as scroll, button clicks, or slider events [MHRL*17].

Delay is the time to start a transition relative to the trigger point. Zero delay coincides with an immediate start of the animation.

Duration is the amount of time to complete a transition. Duration is defined in relation to the animation's start point as the amount of time that transpires until the animation ends.

Easing specifies the speed that a transition progresses at different points in time of the animation. An easing function computes the value of an animated property based on the percentage of time that has progressed in relation to the duration.

4.2. Compositions

The dimension primitives discussed in Section 4.1 are intended to be combined into compositions of animated transitions. In this section we identify commonly employed compositions from our survey of examples. We recognize a set of 10 transition types based on the primitives from the *object*, *graphic* and *data* dimensions. We also point out popular pacing techniques composed from the *timing* primitives in Section 4.1.4. These compositions are not a taxonomy – they do not describe all possible animated transitions. However, these compositions can be combined further to create more complex transitions and animations. We predict future animated data graphic authoring tools will support these compositions.

4.2.1. Transition Types

In our design space, transitions are described by the changes in the *object*, *graphic* and *data* dimensions. When a transition is not specific to an object primitive, it can be applied to multiple object types. We identified a set of 10 commonly employed transition designs found in our survey of examples. It is obvious that these transition types are not exhausive: there are many more possible primitive compositions.

Enter/Exit: (visual presence + data presence) A data-bound object is entirely added or removed from the scene graph. The change occurs in the object's visual presence and data presence. Applicable object types include glyphs, objects, annotations, and axes and legends. An example of enter/exit is the introduction of a new glyph or chart.

Combine/Partition: (data transform + visual presence + configuration) Objects are combined or partitioned based on a nesting data transform and re-arranged by a change in layout configuration.

Each objects' visual presence changes. On aggregate, the attached data tuples are not added or removed from the collection (i.e., data presence), however data tuples are re-attached to objects via data transformation. An example is the partitioning of a bar chart using a data attribute, resulting in a stacked bar chart.

Visual Alteration: (visual attributes) Here an object's visual attributes change, which is specified by the designer and independent from data. The object's data encodings, data transforms, data presence remain unchanged.

Data Encoding Alteration: (data encodings) Data encodings are altered, added, or removed from the data graphic. This change may or may not result in changes in terms of visual presence or visual attribute. For example, if the data value remains unchanged, so does the visual attribute.

Ordering/Re-configuring: (configurations) This transition typically applies to a group, where its layout configuration (graphical dimension) changes, resulting changes in the spatial positions (graphical dimension) of its constituent members. This configuration change may be based on data (e.g. sorting by a data attribute), or may be based on stylistic considerations only.

Highlighting/Filtering: (data queries + visual presence | visual attributes) Objects are visually highlighted or filtered based on the inclusion or exclusion selections defined by a data query. Here the relevant graphical dimension primitives include visual presence and visual attributes, and the relevant data dimension primitive is data queries. Applicable objects include glyphs, objects and annotations.

Data Ticker: (data presence) The visual states cycle through the values of a temporal or nominal data attribute. This attribute is typically orthogonal to the data attributes represented in the graphic. This type of transition applies to glyphs and objects. An example of data ticker is the animated bubble chart over time.

Appear/Disappear: (visual presence) Objects visibility changes in the scene. The main difference between this type and enter/exit is that enter/exit involves data presence as a primitive, but appear/disappear does not. Examples of appear/disappear include the introduction of an annotation, which is not bound to data.

Camera Alteration: (camera + configurations) The camera's configuration such as position or projection properties are altered, resulting in a view change (e.g., panning, zooming, rotating).

Simulated Process: (data transforms) Glyphs animate based on a simulated process, defined by an underlying algorithm or streaming data source.

4.2.2. Pacing Techniques

Here we describe popular pacing techniques based on the *timing* primitives discussed in Section 4.1.4. The following compositions rely on relative timing concepts to build up successive animations into larger compositions. Pacing techniques are designed to assist the audience perceive and apprehend how data graphics change during animated transitions.

Staging segments an object's visually complex transition into sub-transitions, allowing for multiple changes to be easily observed. Staged transitions rely on subsequent sub-transitions to trigger after the previous sub-transition ends. Heer and Robertson [HR07] demonstrated that staged transitions are preferred for visually tracking changes and slightly reduce errors.

Layering is similar to *staging*, as sub-transitions precede after one another. However, layering typically is applied to introduce objects of a data graphic in a piece-meal fashion. Also, momentary pauses or *dwells* are applied between the sub-transitions. Layering follows a formula of sub-transitions triggering after the previous sub-transition ends, with a delay in-between called a *dwell* that allows the audience to apprehend newly introduced objects. According to Schwabish [Sch19], layering is an effective technique to manage the audience's attention as you ask them to follow the progression of presented information.

Staggering is the incremental or distributed delay of animations' start times across a collection of objects' sub-transitions. Typically staggering is applied to data glyphs or axes & legends as the ordering of start times is based on data or visual attributes. The mapping for each objects' animation start time can be defined by sequential, linear, ordinal, or cardinal functions from a single trigger point. Staggering only relates to the start time of an objects' sub-transition; sub-transitions are not required to precede each other. Our formulation of staggering differs from that of Chevalier et al. [CDF14], where objects move after the previous object comes to rest. They found that staggering has negligible, or even negative, impact on an observer's ability to track multiple objects.

Looping is a cyclic succession of transitions that loops back to the start. Sub-transitions occur one after another in a cycle. Loops are similar in form to *staging and layering*, however the initial sub-transition animates after the last transition - thus closing the loop. This repetitive technique is often used to display cyclic temporal data. According to Lena Groeger of ProPublica: "looping makes us notice differences because our attention can shift around to different places" [Gro15].

5. Semi-Structured Ideation Study

To understand how designers think about and approach authoring animated data graphics, we conducted an ideation study with participants possessing experience in graphic, visualization, and animation design. We use the three paradigms from Section 2.1 as inspirational exemplars and asked participants to discuss how they would author six given animated transitions *conceptually*. We seek to answer the following three research questions:

- Q1: Among the graphical, temporal, and data components of an animated transition, which are more commonly associated with a preferred authoring paradigm?
- Q2: How do the characteristics of an animated transition (e.g., transition type) affect the participants' choice of authoring paradigms?
- Q3: What implications do the participants' preferences have on the design of authoring systems and interfaces?

We instructed the participants to draw from their previous experience designing animated graphics, but also to conceive of computer-assisted concepts that would be useful for manipulating data-bound visualizations and transitions. We chose this semi-structured ideation format over a re-construction task with an existing tool (e.g.,Adobe

After Effects [Ado20a]) because we did not want a particular tool to constrain the participants' thinking. We wanted designers to consider all relevant authoring paradigms. Furthermore, creating animated data graphics with current design tools is time-consuming; an ideation study allows participants to experience a breadth of animated transitions in a reasonable amount of time.

5.1. Tasks

For each task, we asked participants to discuss how they would author a given animated transition. We selected 12 animated transition instances from our survey, these instances originate from the following 4 examples: *The Timing of Baby Making* by Amber Thomas [Tho17]; *Twenty Years of the NBA Redrafted* by Russell Goldenberg [Gol17]; *I'm Not Feeling Well* by Gabriel Gianordoli [GSBS19]; *A visual introduction to machine learning* by Stephanie Yee and Tony Chu [YC15]. We chose these animated transition instances based on their coverage of the design space from Section 4 and breadth of graphical complexity. The 12 instances represent the 10 transition types from our design space. We asked each participant to complete 6 tasks (6 of 12 total).

5.2. Participants

We recruited 14 designers (10 female, 4 male) from the Atlanta metropolitan area. On average the participants were 27.9 years old (min = 22; max = 39). Most of them are academics (2 undergraduate students, 10 graduate students), and 2 participants are working professionals. Their years of experience in graphic design is: less than 1 yr = 1 (7.1%); 1-2 yrs = 4 (28.6%); 2-5 yrs = 6 (42.9%); 5-8 yrs = 0 (0%); more than 8 yrs = 3 (21.4%). The distribution of their experience in creating charts, infographics, and data visualizations is: none = 1 (7.1%); less than 1 yr = 2 (14.3%); 1-2 yrs = 7 (50.0%); 2-5 yrs = 2 (14.3%); 5-8 yrs = 0 (0%); more than 8 yrs = 2 (14.3%). Participants reported using the following categories of tools to create data graphics: vector editors = 11 (78.6%); presentation tools = 11 (78.6%); spreadsheets = 8 (57.1%); visualization software = 7 (50.0%); programming toolkits = 7 (50.0%); image editing tools = 6 (42.9%); infographic tools = 1 (7.1%).

The prevalence of participants' experience in creating professional animations is: none = 0 (0.0%); less than 1 yr = 10 (71.4%); 1-2 yrs = 4 (28.6%). When asked about the infrequence of their experience creating animated graphics, many participants responded in regards to their professional experience creating animations, but they had additional years of experience in the classroom. Participants reported creating the following categories of animations: UX animation = 8 (57.1%); interactive visualizations = 6 (42.9%); effects animation = 4 (28.7%); 3D animation = 4 (28.6%); character animation = 3 (21.4%); motion graphics = 3(21.4%); data narratives = 2 (14.3%); stop motion = 0 (0.0%). Participants reported using the following tools for creating animations: Adobe After Effects = 9 (64.3%); InVision = 6 (42.9%); Microsoft PowerPoint / Keynote = 5 (35.7%); Blender = 3 (21.4%); Autodesk Maya = 2 (14.3%); Programming Toolkits = 2 (14.3%).

5.3. Experimental Setup

The study took place in a laboratory setting, with each session taking 1.5 hours. Sessions were audio recorded and participants interacted with a 2880x1800 screen laptop. Participants first watched an 8-minute informational video that explains fundamental data visualization concepts and the three authoring paradigms. The informational video highlights the differences between the three paradigms: keyframing as declaring a time and property for the system to tween between states; procedural as a system of rules that updates graphics based on specified parameters or events for each frame of the animation; and presets & templates as predetermined transitions that are applied to graphics using relative timing.

During the task portion, participants worked on 6 (out of 12) total animated transition instances originating from 2 (out of 4) examples. For each task, participants first familiarized themselves with the example on a Chrome web browser for 3-5 minutes to understand the context of the animations. Participants then completed a *worksheet-guided analysis* and an *authoring ideation task*.

Worksheet-Guided Analysis: We presented each animation as a slowed-down screen-capture via QuickTime Player. Upon viewing the animation, we prompted the participant to fill-out a worksheet. The worksheet assists in building and externalizing participants' understanding of the animated transition in terms of the *object*, *graphics*, *data*, and *timing* dimensions proposed in Section 3. The purpose of the worksheet was not to test participants' abilities to analyze animated transitions. Therefore, participants were provided answers upfront (during pilot studies, data change and duration were difficult to discern from a screen-capture alone), and corrected if they made an error during analysis. Completing the worksheet was a prerequisite for the participant to discuss how they would *conceptually* author the animated transition.

Authoring Ideation Task: We asked the participants how they would approach authoring the animated transition instance in a visual authoring system rather than textual programming. We prompted them to consider three authoring paradigms presented in the informational video, and encouraged them to think creatively beyond the paradigms if necessary. We also asked the participants to think-aloud; drawing on past experiences with familiar tools to imagine useful system concepts. We encouraged participants to sketch ideas on paper. Based on their initial ideas and sketches, we asked follow-up questions and urged participants to consider deeply how they would author each component of the animated data graphic.

Upon completing the 6 tasks participants answered debrief questions to summarize their ideas, provide additional thoughts, and reflect on their past design experiences.

5.4. Analysis

We analyzed the audio recordings from each task. We did not include the worksheet-guided analysis recordings, as they do not include discussions on authoring. In total, we recorded 84 completed tasks (6 tasks $\times$ 14 participants). We transcribed each task, including in-situ references to participants' sketches.

We sought to answer our three research questions by coding

TASK ID	EXAMPLE NAME	TRANSITION TYPE	Keyframe KF	Procedural PD	Presets & Templates PT
CONDITION A					
A1	NBA Redraft	Ordering/Re-configuring	85.7%	28.6%	42.9%
A2	NBA Redraft	Highlighting/Filtering	85.7%	0%	85.7%
A3	NBA Redraft	Data Encoding Alteration	71.4%	42.9%	57.1%
A4	Baby Making	Appear/Disappear	14.3%	0%	100%
A5	Baby Making	Data Ticker	71.4%	42.9%	57.1%
A6	Baby Making	Combine/Partition	100%	0%	42.9%
CONDITION B					
B1	Not Feeling Well	Data Encoding Alteration	71.4%	14.3%	28.6%
B2	Not Feeling Well	Camera Alteration	57.1%	71.4%	28.6%
B3	Visual Intro to ML	Enter/Exit	85.7%	57.1%	42.9%
B4	Visual Intro to ML	Data Encoding Alteration	100%	71.4%	28.6%
B5	Visual Intro to ML	Visual Alteration	71.4%	71.4%	28.6%
B6	Visual Intro to ML	Simulated Process	42.9%	100%	0%
ALL TASKS					
			71.4%	41.7%	45.2%

Table 1: *Distribution of 12 animated transition instances employed across conditions A & B. Results show percentage of participants that described an authoring paradigm for each task.*

the transcripts as follows: First, code the authoring paradigm expressed by the participant (procedural animation, keyframe animation, and/or presets & templates); *Q1* - code the animated data graphic by the *object*, *graphic*, *data* and *timing* dimensions from Section 4; *Q2* - code each animated transition instance based on the transition types from Section 4; *Q3* - apply open coding to identify user interface ideas and system features proposed by the participant.

We only coded responses relevant to authoring. Many utterances were not relevant: participants often asked questions about the data graphic or the task at hand; some utterances were unclear on authoring intent; while others were part of the participant's attempt to formulate their thoughts. Two authors individually coded transcripts from 4 sessions (conditions A & B). We measured the inter-coder agreement on authoring paradigm codes using Cohen's Kappa (K) [Coh60]. Based on Kandis and Loch's scale [LK77] our codes are in "substantial agreement" as K = 0.69. Once in agreement of codes, the primary author completed the coding for all 14 sessions.

5.5. Results

To help explain the results of our analysis, we denote the three authoring paradigms with the following shorthand: [keyframe (KF), procedural (PD), presets & templates (PT)]. Overall, across all authoring tasks (84 total, 6 tasks $\times$ 14 participants), the participants described authoring with keyframe animation most frequently [KF: 71.4%], followed by presets & templates [PT: 45.2%], and finally procedural animation [PD: 41.7%].

Q1: Which animation components are more commonly associated with a preferred authoring paradigm?

Co-occurrence analysis of animation components (e.g., object, timing, data) identified the elements that participants frequently modify when authoring with either of the three authoring paradigms. In this portion of the analysis, percentages are based on a denominator equal to the total number of tasks discussed by participants for each authoring paradigm (e.g., 49 is the total number of tasks where participants discussed glyphs and keyframe animation; 60 is the total number of tasks where participants discussed keyframe animation; therefore during 49/60 or 81.6% of tasks, participants discussed authoring with keyframe animation using glyphs).

Object Dimension:
Data Glyphs: Participants most frequently employed procedural animation and keyframe animation when working with glyphs [**KF: 88.6%**, **PD: 81.7%**, PT: 23.8%]. Participants described procedural conditions or rules that would apply for all glyphs. Similarly, they also described keyframes as being repeated for glyphs. However, some participants struggled to come up with the "apply to all glyphs" concept – P7 claimed: "*If I don't have so many objects - maybe only 10 objects are moving. I think I'm comfortable with After Effects, but like this one, I guess there are hundreds of objects moving at the same and it must be complicated.*" This is a fair comment if the participant must manually create hundreds of keyframes or procedures for hundreds of glyphs. However, the generative concept of "apply to all glyphs" or "repeat for glyphs" that many participants described alleviates this manual burden.

Visual States: Participants also considered the entire static graphic at rest – they expressed how they would create animations from visual states more frequently with keyframe animation and presets & templates [**KF: 48.3%**, PD: 20%, **PT: 42.1%**]. Participants described how keyframe animation works well for setting two different data graphics as frames and allowing the system to link the glyphs by data, interpolating the properties that transition between each state. Participants also frequently described applying presets & templates to frames in a "slide show" or "page viewer" interface. Participants described how presets & templates could be applied to an entire static frame to create a transition.

Groups: Furthermore, participants described groups as a way to structure animations, most frequently they described groups when working with keyframe animation or procedural animation [**KF: 16.7%**, **PD: 14.3%**, PT: 5.3%]. Groups were typically created via procedural statements such as "group glyphs by data attribute". Participants desired to apply keyframes to groups of data glyphs to achieve staging – for example P2 explained how they would first "*create different groups for both blue and green [rectangles]*" to rotate them all together and then transition the rectangles y-position. Participants also desired groups as a means to order an otherwise cluttered timeline interface.

Timing Dimension: We considered how participants authored timing effects (e.g., delay, duration, easing) in relation to the three authoring paradigms. We found that participants often expressed the need to create functions based on data or visual attributes that specify delay functions for glyphs [KF: 43.3%, **PD: 68.6%**, PT: 44.7%]. We also found that participants described presets & templates as a method to stagger delay based on data or visual attributes, P4 referred to each glyph animating after the other glyph as the "domino effect." Some participants described the process of creating staging frequently with keyframe animation and presets & templates [**KF: 11.7%**, PD: 8.6%, **PT: 18.4%**]. Participants described adding keyframes for visual attributes or encodings of glyphs to stage a complex transition; while other participants described the relational timing of presets & templates (e.g.,“animate after”, “animate with”) as a straight-forward method to stage transitions after one another. When it comes to the duration of transitions, participants did not have a clear preference for authoring paradigms [KF: 30%, PD: 28.6%, PT: 34.2%]. Finally, participants seldom described easing functions [KF: 5%, PD: 2.9%, PT: 2.6%]. This could be related to a lack of identifying easing in the worksheet-guided analysis.

Data Dimension: We associated how participants described different aspects of datasets (e.g., data tuples, data attributes, data transforms) with each of the authoring paradigms. We found that participants most frequently described procedural statements that altered the data tuples bound to glyphs [KF: 23.3%, **PD: 34.3%**, PT: 10.5%]. Participants frequently described procedural statements to alter the data transforms of the glyphs to achieve a change in bound data tuples. Therefore, data transforms were more frequently associated with procedural animation than other authoring paradigms [KF: 13.3%, **PD: 28.6%**, PT: 10.5%]. Participants most frequently used data attributes when describing procedural rules or statements [KF: 50%, **PD: 71.4%**, PT: 44.7%]. These procedural rules include the previously mentioned delay functions, pivoting data transforms by a data attribute, or creating rules for data encodings. Data attributes were also expressed as ingredients for the other two authoring paradigms. Participants created keyframes by altering a data encoding’s data attribute. Participants also discussed creating pre-defined timing effects based on a selected data attributes.

Q2: How do the characteristics of an animated transition (e.g., transition type) affect the participants’ choice of authoring paradigms?

The results show that participants have preferred authoring paradigms for certain transition types proposed in Section 4. In Table 1 we compare authoring paradigms across each task (row). The columns of Table 1 describe transition types and distribution of authoring paradigms expressed by participants. The rows do not total to 100% as participants expressed multiple authoring paradigms for a single task. In this portion of the analysis, percentages are based on a denominator of 7 participants completing each task.

Participants discussed creating appear/disappear transitions using presets & templates more than any other paradigm [KF: 14.3%, PD: 0%, **PT: 100%**]. Participants cited the “lack of data bindings" and “small number of graphic objects" as justification for using presets & templates. They also remarked that such transitions are similar to the appear/disappear presets from familiar tools. Participants described creating simulated processes with procedural authoring most frequently [KF: 42.9%, **PD: 100%**, PT: 0.0%]. Participants claimed that state-based simulations would most easily translate to procedural rules. According to participants, keyframe animation best suits combine/partition transitions [**KF: 100%**, PD: 0%, PT: 42.9%]. Keyframe animation provides a visual timeline to control the combination of objects between these key states. Participants also imagined that keyframing could link partial objects to combined objects based on underlying data (and vice versa). Participants favored using keyframe animation or presets & templates for highlighting/filtering transitions [**KF: 85.7%**, PD: 0%, **PT: 85.7%**].

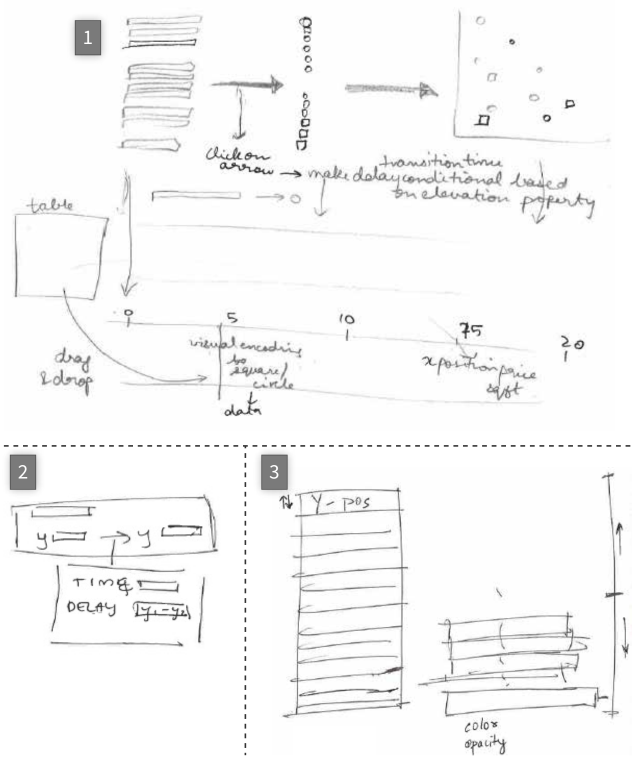

Figure 4: *Participants’ user interface sketches for an animated data graphic authoring tool. (1) P12’s interface combines storyboarding and timeline views, (2) P5’s concept for modifying a transition’s time components such as delay functions, (3) P10’s data table supports ordering the delay of glyphs’ transitions by visual attributes such as y-position*

For all other transition types, participants did not have an overwhelming preference for an authoring paradigm. The only cases where 0 participants described using an authoring paradigm for an transition types were: procedural animation for highlighting/filtering, appear/disappear, and combine/partition [PD: 0%]; and presets & templates for simulated processes [PT: 0%]. Among those remaining transition types, participants described authoring at least once for the three paradigms.

Q3: What implications do the participants’ preferences have on user interface design and system features?

We asked participants to comment on useful user interface designs and system concepts for an authoring system during each task. Participants often explained how interfaces from familiar tools or completely novel concepts would be useful for authoring animated data graphics. In this section we analyzed the frequency of user interface designs and system concepts described by the 14 participants. Percentages are based on a denominator of 14 (14 total participants).

Storyboards: 9 of 14 participants [64.3%] commented on how a slide show or storyboarding interface would be useful for authoring animated data graphics. These storyboards serve as representations of high-level story points for quick specification, sharing, and preview. As seen at the top of P12's sketch in Figure 4.1 – the stages of an animation are displayed as storyboards with arrows between each storyboard that represent the animated transition between each stage. P12 envisions that "transition arrows" are selected to edit animation properties of that transition. Beyond smaller staged transitions, participants desired to create and layout slides or storyboards for each crucial story point in their larger narrative. P4 and P10 cited the need to quickly iterate on these key story points at a high-level and share storyboard ideas with colleagues or stakeholders before investing time to author animations between story points. 10 participants [71.4%] explained that animation previews would be helpful to compare and contrast potential animation design options. According to P14: "*Previews would help. When you set an effect...you can see the preview in the background...So that would be helpful because [the system] could create a GIF. You could watch it happen [and] decide if you need to modify [the animation] in real time.*"

Timelines: 8 of 14 participants [57.1%] described how they would want to use a timeline interface to author animations. This correlates with the participants favoring keyframe animation for the majority of tasks. As seen at the bottom of P12's sketch in Figure 4.1 – the stages of a transition could be created as keyframes on a timeline. Figure 4.1 depicts P12's combined storyboard and timeline interface design. In this design, storyboards are linked together by transitions and further specification of transitions occurs in a timeline interface – this dual interfaces appears in related prototyping tools such as InVision [Inv20]. We hypothesize that that a similar dual interface would be useful for authoring animated data graphics. Participants had two approaches for creating keyframes: by transforming the preexisting data graphic in the system (e.g., changing a visual attribute or visual encoding), or by importing static data graphics created in another tool and relying on the system to link the two static frames by association.

Timing: Participants considered many different options to design the timing of animations. 7 participants [50.0%] wanted to order glyphs in a list interface to specify delay based on data or visual attributes as seen in P10's sketch in Figure 4.3. 6 participants [42.9%] described using a formula editor interface to set duration or delay based on computed functions from data or visual attributes such as P5's drawing in Figure 4.2. While 6 participants [42.9%] described the need to select timing effects from a list of presets & templates such as the "domino effect" described by P4 to stagger glyphs' delay. 4 participants [28.6%] described using relational timing methods such as "animate after" or "animate with" to stage animations.

Working with data: 5 participants [35.7%] described the need to have a data table interface for inspecting data. All but one participant [92.9%] described the system concept to link glyphs by their underlying data tuples or data transforms. Participants also described the desire to have system concepts to assist with data-driven generation: 11 participants [78.6%] described auto-generation of visual encodings, 6 participants [42.9%] described the need for the system to provide highlighting or filtering options based on a selected data attribute, while 6 participants [42.9%] desired the ability to sort glyphs by data attributes, and 5 participants [35.7%] described grouping glyphs by a categorical data attribute.

6. Discussion

The study results show that keyframe animation is an authoring paradigm preferred by many participants in a majority of tasks, with the exception of three transition types: appear/disappear, simulated process, and camera alteration. P11 remarked that "*I may not necessarily use the term keyframe all the time but it's how I think*". A likely reason is the participants' experience and familiarity with keyframing tools. 9 of 14 participants were familiar with Adobe After Effects; 6 of 14 reported using InVision. Participants also indicated that they preferred keyframe animation due to its ability to define transitions on top of a static data graphic.

When data becomes an integral part of animated graphics, however, current keyframing tools become inadequate. First, animated data graphics often involve hundreds or more data glyphs. It is difficult to keep track of and operate on all of them. "*I think the hard thing would be when you have so many elements animated in one scene or one keyframe. That would be really hard to keep tracking, make sure all the delay or all of the duration time makes sense.*." (P6). Second, it takes a significant amount of time and effort to specify animation behaviors of glyphs based on data. In completing task 4, P4 mentioned "*if you do kind of like a trim path animation in After Effects it takes an immense amount of time to do. Umm, because each of the shapes that you're trying to animate have to be individual shapes and they all have their own individual trim path property*." It is essential that future authoring tools for animated data graphics support functionalities such as automatic mapping of data to animation properties, automatic transition types like data ticker, and batch visual alteration.

While keyframe animation may be the most popular paradigm, the other two paradigms are indispensable for certain tasks. For example, in task 1, a group of rectangle glyphs representing the drafted NBA players animate to update their positions as their vertical positions' data encodings change. The animations do not happen simultaneously for all the glyphs; instead, one glyph starts animating after another. This kind of behavior may be tedious to specify at the glyph level under the keyframe paradigm, and P1 preferred to describe this behavior as the "domino effect" and wished there were such an animation template to be readily applied.

Multiple participants also noted the need to combine multiple paradigms so that the authoring tools strike a balance between ease of use and control. In particular, the presets & templates paradigm is acknowledged to be easy to use, but it also implies a lack of precise control: "*And so I think that After Effects allows you a maximum control. But then the downside is you can't [create] as fast as you would with Keynote. And so I think I want a tool that is a combination of both of those [concepts].*" (P4). To support such needs, the participants provided inspirational design ideas for user interfaces. In Section 5.5, the storyboard interface and the timeline interface may be tightly coupled so that users can easily navigate between these two paradigms.

Finally, some of the animations are still best expressed in the procedural paradigm: at the component level, the results show that the participants preferred the procedural paradigm for timing components such as delay function based on data or visual attributes, or operations related to data components; at the composition level, transition types such as simulated process are easier to describe procedurally as well. The procedural paradigm is closely related to programmatic thinking, even if the interface may not be in the form of a programming language: "*if we are still using the thinking of program then we're still there. We are not far from programming*" (P7). We may devise novel interfaces for the procedural paradigm without programming, but the ability to think procedurally may be essential in the authoring of animated data graphics.

7. Conclusion and Future Work

We presented the results of an ideation study that examines users' preferences of authoring paradigms for animated data graphics. This study is based on an analysis of *object*, *graphic*, *data* and *timing* primitives and their compositions in the design space of animated data graphics. We found that keyframe animation is the most preferred paradigm, but certain animation components and transition types are better described procedurally or in presets & templates.

In our future work, we aim to design and implement an animation authoring tool that offers a combination of these paradigms to help authors balance between ease of use and expressive control. We envision this authoring tool should allow authors to produce animated data graphics, without writing textual programming. Furthermore, this authoring system should allow authors to construct narrative sequences of animated data graphics and share them on the web.

8. Acknowledgments

This work was supported in part by Adobe Research.

References

[Ado19] ADOBE INC.: Expression basics: Learn expression basics to link animations in Adobe After Effects, Nov 2019. URL: http://helpx.adobe.com/after-effects/using/expression-basics.html. 2

[Ado20a] ADOBE INC.: Adobe After Effects, 2020. URL: http://www.adobe.com/products/aftereffects.html. 1, 2, 3, 7

[Ado20b] ADOBE INC.: Adobe Character Animator, 2020. URL: http://www.adobe.com/products/character-animator.html. 3

[Ado20c] ADOBE INC.: Adobe XD, 2020. URL: http://www.adobe.com/products/xd.html. 1, 2

[AHRL*15] AMINI F., HENRY RICHE N., LEE B., HURTER C., IRANI P.: Understanding Data Videos: Looking at Narrative Visualization through the Cinematography Lens. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (2015), CHI '15, ACM, pp. 1459–1468. doi:10.1145/2702123.2702431. 3

[AMM*08] AIGNER W., MIKSCH S., MÜLLER W., SCHUMANN H., TOMINSKI C.: Visual Methods for Analyzing Time-Oriented Data. *IEEE Transactions on Visualization and Computer Graphics 14*, 1 (2008), 47–60. doi:10.1109/TVCG.2007.70415. 1

[App20] APPLE INC.: Keynote - Apple, 2020. URL: http://www.apple.com/keynote/. 1

[ARL*17] AMINI F., RICHE N. H., LEE B., MONROY-HERNANDEZ A., IRANI P.: Authoring Data-Driven Videos with DataClips. *IEEE Transactions on Visualization and Computer Graphics 23*, 1 (2017), 501–510. doi:10.1109/TVCG.2016.2598647. 3, 4

[Ble02] BLENDER: Blender - Free and Open 3D Creation Software, 20202. URL: http://www.blender.org. 2

[BOH11] BOSTOCK M., OGIEVETSKY V., HEER J.: D^3 Data-Driven Documents. *IEEE Transactions on Visualization and Computer Graphics 17*, 12 (2011), 2301–2309. doi:10.1109/TVCG.2011.185. 2, 3

[Bre20] BREMER N.: Visual Cinnamon, 2020. URL: http://www.visualcinnamon.com/. 4

[Bro88] BROWN M. H.: *Algorithm Animation*. MIT Press, Cambridge, MA, USA, 1988. 2

[CDF14] CHEVALIER F., DRAGICEVIC P., FRANCONERI S.: The Not-so-Staggering Effect of Staggered Animated Transitions on Visual Tracking. *IEEE Transactions on Visualization and Computer Graphics 20*, 12 (2014), 2241–2250. doi:10.1109/TVCG.2014.2346424. 6

[Cha18] CHALBI A.: *Understanding and designing animations in the user interfaces*. PhD thesis, Université de Lille, Apr 2018. URL: https://hal.archives-ouvertes.fr/tel-01881889. 3

[Cla19] CLARK D.: Why data visualisation needs a play button, 2019. URL: https://flourish.studio/2019/02/07/audio-talkie-visualisation-data-stories/. 3

[Coh60] COHEN J.: A coefficient of agreement for nominal scales. *Educational and Psychological Measurement 20*, 1 (Apr 1960), 37–46. doi:10.1177/001316446002000104. 8

[CRP*16] CHEVALIER F., RICHE N. H., PLAISANT C., CHALBI A., HURTER C.: Animations 25 Years Later: New Roles and Opportunities. In *Proceedings of the International Working Conference on Advanced Visual Interfaces* (Jun 2016), AVI '16, ACM, pp. 280–287. doi:10.1145/2909132.2909255. 1, 3

[DCL08] DAVIS R. C., COLWELL B., LANDAY J. A.: K-sketch: A 'Kinetic' Sketch Pad for Novice Animators. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (2008), CHI '08, ACM, pp. 413–422. doi:10.1145/1357054.1357122. 1

[Few06] FEW S.: *Information Dashboard Design: The Effective Visual Communication of Data*. O'Reilly Media, Inc., 2006. 2

[Fis10] FISHER D.: Animation for Visualization: Opportunities and Drawbacks. In *Beautiful Visualization Edition*. O'Reilly Media, Apr 2010. URL: https://www.microsoft.com/en-us/research/publication/animation-for-visualization-opportunities-and-drawbacks/. 3

[Gol17] GOLDENBERG R.: Twenty Years of the NBA Redrafted, Mar 2017. URL: http://pudding.cool/2017/03/redraft/. 7

[Goo20] GOOGLE NEWS LAB: Visualizing Google data, 2020. URL: http://trends.google.com/trends/story/US_cu_6fXtAFIBAABWdM_en. 4

[Gro15] GROEGER L.: That's the Power of Loops, 2015. URL: http://www.youtube.com/watch?v=zd0YQAgu3dI. 6

[GSBS19] GIANORDOLI G., SALABERRY L., BIERNATH A., SYAM U.: I'm Not Feeling Well, 2019. URL: http://www.imnotfeelingwell.com/. 7

[Hal20] HALLORAN N.: Neil Halloran, 2020. URL: http://www.neilhalloran.com/. 4

[HR07] HEER J., ROBERTSON G.: Animated Transitions in Statistical Data Graphics. *IEEE Transactions on Visualization and Computer Graphics 13*, 6 (2007), 1240–1247. doi:10.1109/TVCG.2007.70539. 3, 4, 6

[HS93] HUDSON S. E., STASKO J. T.: Animation Support in a User Interface Toolkit: Flexible, Robust, and Reusable Abstractions. In *Proceedings of the ACM Symposium on User Interface Software and Technology* (1993), UIST '93, ACM, pp. 57–67. doi:10.1145/168642.168648. 1

[Inv20] INVISIONAPP INC.: Invision, 2020. URL: http://www.invisionapp.com/. 1, 10

[Kah17] KAHN J.: Rekapi: A JavaScript Keyframe Library, Dec 2017. URL: http://jeremyckahn.github.io/rekapi/doc/. 2

[Kan20] KANTAR: Information is Beautiful Awards, 2020. URL: http://www.informationisbeautifulawards.com/. 4

[KCG*14] KAZI R. H., CHEVALIER F., GROSSMAN T., ZHAO S., FITZMAURICE G.: Draco: Bringing Life to Illustrations with Kinetic Textures. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (2014), CHI '14, ACM, pp. 351–360. doi:10.1145/2556288.2556987. 1, 2, 3

[KCGF14] KAZI R. H., CHEVALIER F., GROSSMAN T., FITZMAURICE G.: Kitty: Sketching Dynamic and Interactive Illustrations. In *Proceedings of the ACM Symposium on User Interface Software and Technology* (2014), UIST '14, ACM, pp. 395–405. doi:10.1145/2642918.2647375. 1

[KHBB16] KRAMER J.-P., HENNINGS M., BRANDT J., BORCHERS J.: An Empirical Study of Programming Paradigms for Animation. In *IEEE/ACM Cooperative and Human Aspects of Software Engineering (CHASE)* (May 2016), IEEE, pp. 58–61. doi:10.1109/CHASE.2016.020. 3

[Kil19] KILN ENTERPRISES LTD: Flourish Studio, 2019. URL: http://flourish.studio. 2, 3

[KS02] KERREN A., STASKO J. T.: Chapter 1: Algorithm Animation. In *Software Visualization* (Berlin, Heidelberg, Germany, 2002), Diehl S., (Ed.), Springer-Verlag, pp. 1–15. 2

[LK77] LANDIS J. R., KOCH G. G.: The Measurement of Observer Agreement for Categorical Data. *Biometrics 33*, 1 (1977), 159–174. doi:10.2307/2529310. 8

[LPR19] LIN PEDERSEN T., ROBINSON D.: A Grammar of Animated Graphics: gganimate, 2019. URL: http://gganimate.com/. 3

[LTW*18] LIU Z., THOMPSON J., WILSON A., DONTCHEVA M., DELOREY J., GRIGG S., KERR B., STASKO J.: Data Illustrator: Augmenting Vector Design Tools with Lazy Data Binding for Expressive Visualization Authoring. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (2018), CHI '18, ACM, pp. 1–13. doi:10.1145/3173574.3173697. 3

[MHRL*17] MCKENNA S., HENRY RICHE N., LEE B., BOY J., MEYER M.: Visual Narrative Flow: Exploring Factors Shaping Data Visualization Story Reading Experiences. *Computer Graphics Forum 36*, 3 (2017), 377–387. doi:10.1111/cgf.13195. 5

[Mic20] MICROSOFT CORP.: Microsoft Powerpoint, 2020. URL: http://products.office.com/en-us/powerpoint. 1, 3

[New18] NEW YORK TIMES: 2018: The Year in Visual Stories and Graphics, 2018. URL: http://www.nytimes.com/interactive/2018/us/2018-year-in-graphics.html. 4

[Pri20] PRINCIPLE: Principle, 2020. URL: http://www.principleformac.com/. 1

[Pud20] PUDDING, THE: The Pudding Archives, 2020. URL: http://pudding.cool/archives/. 4

[RF06] REAS C., FRY B.: Processing: programming for the media arts. *AI & Society 20*, 4 (Aug 2006), 526–538. doi:10.1007/s00146-006-0050-9. 2, 3

[RHR16] ROBERTS J. C., HEADLEAND C., RITSOS P. D.: Sketching Designs Using the Five Design-Sheet Methodology. *IEEE Transactions on Visualization and Computer Graphics 22*, 1 (2016), 419–428. doi:10.1109/TVCG.2015.2467271. 3

[Sch19] SCHWABISH J.: 4 Observations on Animating Your Data Visualizations, Jul 2019. URL: https://link.medium.com/OBs05OgeD3. 6

[Ste20] STEFANER M.: Truth & Beauty, 2020. URL: http://truth-and-beauty.net/. 4

[Tho17] THOMAS A.: The Timing of Baby Making, May 2017. URL: https://pudding.cool/2017/05/births/. 7

[Tou20] TOUCH DESIGNER: TouchDesigner by Derivative, 2020. URL: http://www.derivative.ca. 2

[Tum20] TUMULT: Tumult Hype 4.0, 2020. URL: http://www.tumult.com/hype/. 1

[WHC15] WALNY J., HURON S., CARPENDALE S.: An Exploratory Study of Data Sketching for Visual Representation. *Computer Graphics Forum 34*, 3 (2015), 231–240. doi:10.1111/cgf.12635. 3

[Wic09] WICKHAM H.: *ggplot2: Elegant Graphics for Data Analysis*. Springer, 2009. 3

[WSC17] WEXLER S., SHAFFER J., COTGREAVE A.: *The Big Book of Dashboards: Visualizing Your Data Using Real-World Business Scenarios*. John Wiley & Sons, 2017. 2

[Wu20] WU S.: Shirley Xueyang Wu, 2020. URL: http://sxywu.com/. 4

[YC15] YEE S., CHU T.: A visual introduction to machine learning, Jul 2015. URL: http://www.r2d3.us/visual-intro-to-machine-learning-part-1/. 7

[ZS03] ZONGKER D. E., SALESIN D. H.: On Creating Animated Presentations. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2003), SCA '03, Eurographics Association, pp. 298–308. doi:10.5555/846276.846319. 1