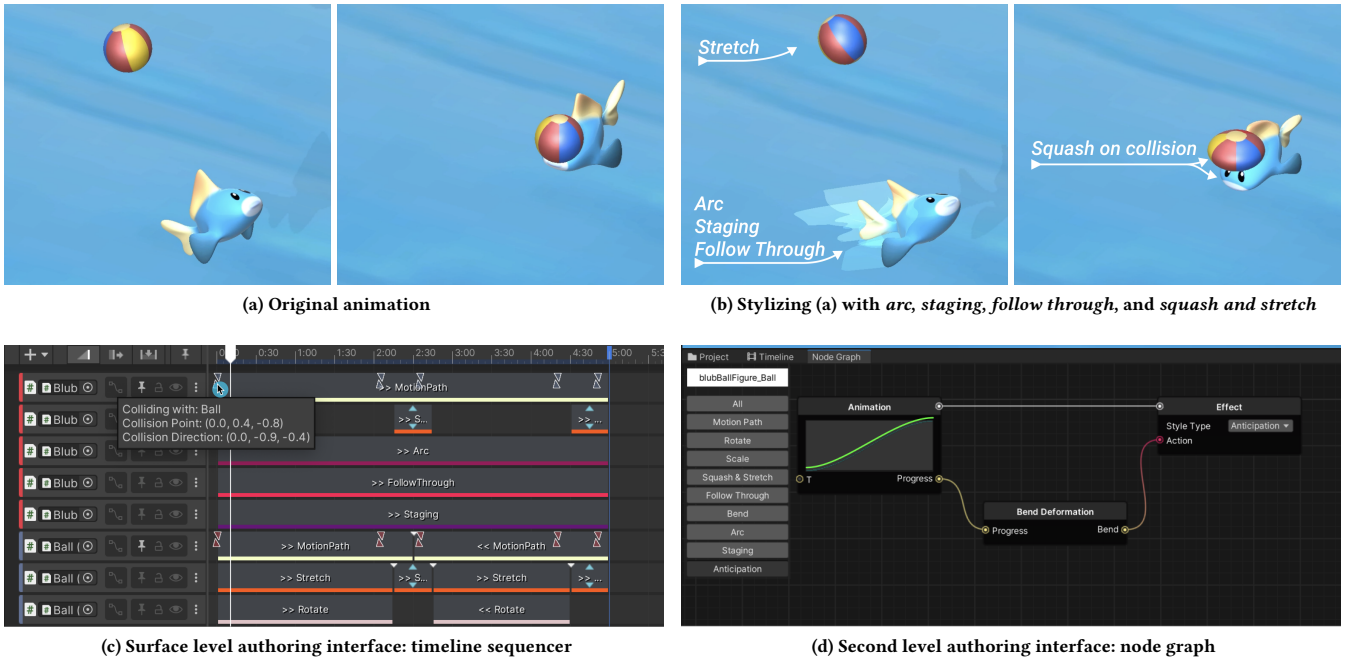
A Layered Authoring Tool for Stylized 3D animations

Jiaju Ma
Brown University, Adobe Research,
Rhode Island School of Design
jiaju_ma@alumni.brown.edu

Li-Yi Wei
Adobe Research
review@liyiwei.org

Rubaiat Habib Kazi
Adobe Research
rhabib@adobe.com



(a) Original animation

(b) Stylizing (a) with *arc, staging, follow through,* and *squash and stretch*

(c) Surface level authoring interface: timeline sequencer

(d) Second level authoring interface: node graph

Figure 1: *Our stylized 3D animation system.* The layered user interface features a timeline sequencer (c) in which users can add stylization effects as additional channels for the corresponding objects (a), either individually or as groups (b). The effects can be added from a library of preset stylizations or further customized via a node graph (d).

ABSTRACT

Guided by the 12 principles of animation, stylization is a core 2D animation feature but has been utilized mainly by experienced animators. Although there are tools for stylizing 2D animations, creating stylized 3D animations remains a challenging problem due to the additional spatial dimension and the need for responsive actions like contact and collision. We propose a system that helps users create stylized casual 3D animations. A layered authoring interface is employed to balance between ease of use and expressiveness. Our surface level UI is a timeline sequencer that lets users add preset stylization effects such as squash and stretch and follow through to plain motions. Users can adjust spatial and temporal parameters to fine-tune these stylizations. These edits are propagated to our node-graph-based second level UI, in which the users can create custom stylizations after they are comfortable with the surface level UI. Our system also enables the stylization of interactions among multiple objects like force, energy, and collision. A pilot user study has shown that our fluid layered UI design allows for both ease of use and expressiveness better than existing tools.

CHI '22, April 29-May 5, 2022, New Orleans, LA, USA

ACM ISBN 978-1-4503-9157-3/22/04…$15.00
https://doi.org/10.1145/3491102.3501894

CCS CONCEPTS

• **Computing methodologies** → **Animation**; • **Human-centered computing** → **Graphical user interfaces**.

KEYWORDS

stylization, animation, 3D

ACM Reference Format:
Jiaju Ma, Li-Yi Wei, and Rubaiat Habib Kazi. 2022. A Layered Authoring Tool for Stylized 3D animations. In *CHI Conference on Human Factors in Computing Systems (CHI '22), April 29-May 5, 2022, New Orleans, LA, USA.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3491102.3501894

1 INTRODUCTION

Stylization has long been applied to traditional animations to give them a sense of life and expressiveness [55], based on the 12 principles of animation that have been serving as the guidelines for both 2D and 3D animations [34]. However, these guidelines still require sufficiently high artistic and technical skills to achieve the desired effects. Although techniques have been proposed to reduce such skill and labor barriers for 2D animations [33], there is a lack of similar solutions for creating stylized animations in the 3D domain to our best knowledge. When compared to its 2D counterpart, 3D animation can be more difficult to teach, learn, and create [23, 26] due to the technical hurdles of 3D computer graphics [23] and the complexity of current 3D animation software [48]. Creating 3D animations in a stylized manner is even more demanding, as it requires the animator "to have both a flair for timing, weight, acting and performance as well as a core technical understanding" [48]. Moreover, 3D animations are often deployed in settings like games and other interactive applications that expect responsive actions like contact and collision [12, 43] that require precise timing and coordination among multiple objects.

When designing creativity support tools, it is important to consider the balance between ease of use and expressiveness [32, 33]. There are three main authoring paradigms among existing animation software [56]: a keyframe-based approach that asks the user to insert keyframes in a timeline (Maya [7], Blender [21]), a procedural approach in which various nodes are connected in a visual programming manner (Houdini [51], TouchDesigner [17]), and a preset-based approach that allows the users to curate from a set of predefined effects (PowerPoint [40], Adobe XD [4]). While the first two approaches in general give the user more control and thus are more expressive, software in those categories is known to be hard to learn for novices [26, 48]. Creating with presets speeds up workflow and is friendly to beginners, but has limited flexibility [32, 56].

We propose a system that helps casual users, especially novices who have some but not extensive domain expertise, create stylized 3D animations. The 3D animation pipeline is complex and composed of many stages: modeling, texturing, rigging, and animating [10]. Fully-fledged 3D animation tools like Maya and Cinema 4D [39] cover the entire pipeline and are designed for feature-length animation productions. We instead concentrate on the animation stage of the pipeline and design for authoring casual animations [18, 31] with applications for domains like marketing, education, and immersive experiences (Figure 9). Thus, our key design focus includes control, expressiveness, and ease of use. Informed by our design study in which 6 professionals in the 3D animation industry were interviewed, we design a **layered authoring interface** that combines the aforementioned three authoring paradigms to fluidly balance between ease of use and expressiveness. On the surface level, the user works with a timeline sequencer similar to those found in existing keyframe-based software (Maya, Blender). However, we remove the keyframing mechanism and instead let users create basic animations and add stylization effects through placing clips grouped by tracks into the sequencer (Figure 1c). To further ease the usage and speed up the workflow, stylizations can be chosen from a preset library that includes motion effects based on animation principles, such as *squash and stretch*, *anticipation*, and *follow through*. The users can further tune additional spatial and temporal parameters of these effects to achieve desired results. These surface level selections and edits are then propagated to our node-graph-based second level authoring interface. Our tool is designed so that the users do not have to work with the node graph UI. Once they become familiar with our tool, they have the option to create custom stylizations not provided in the preset library or change the behavior of existing stylizations by editing the node graph. For example, they can create a custom "wing flapping" effect by using a deformation node and editing its animation curve to control how an object bends. While our surface level timeline sequencer is optimized for ease of use, our node graph interface gives the user more control for extended expressiveness and flexibility. In summary, this layered UI design allows our tool to fluidly move along the ease of use-expressiveness spectrum, depending on the user's familiarity with our tool and their end goals. As a whole, our system allows the stylization of not only individual object effects but also the interactions among multiple objects, such as contact, collision, force, and energy [1, 15].

We evaluate our prototype system through a pilot user study in which both novice and professional animators worked with our system and traditional 3D animation tools. Our results indicate that our tool allows for faster creation of stylized 3D animations while maintaining a good balance between ease of use and expressiveness as we initially set out to achieve. We further demonstrate the capabilities of the system by creating example animations suggested by the participants in the final interviews.

2 RELATED WORK

2.1 Stylized animation authoring

Popularized by Disney movies, the 12 principles of animation [55] have been a cornerstone of traditional hand-drawn animations. Codifying these stylization principles as motion effects for computer animation has gathered significant commercial [2, 51] and research interests [33, 58]. In professional practices, stylized 3D animations are made by animators using commercial tools like Maya and Cinema 4D. While powerful, they are complex to use and require a significant amount of training [48]. A variety of plugins for these tools exist to ease the stylization process, like Squash - Stretch Maker for Maya [24] and Squash & Stretch Kit for Unity [36]. However, these plugins are all special-purposed (support one principle only) and require users to learn their dependent software first.

Prior works also focus on simplifying stylization workflows. MagicalHands [5] maps gestures to animations, while AniMesh [27] retargets motion capture data to rigged 3D models. Cartoon Animation Filter [58] modifies an input motion signal like video to exhibit stylized effects, and Eom et al. [19] similarly proposed a method that generates stylized motions from captured ones. Yet, in these works, the user has limited control over the output and needs to create the input data via other means. Space-Time Sketching moves and deforms 3D characters along drawn curves [22]. Artist-Directed Dynamics [8] and Dynamic Sprites [28] simulates transitions from one character pose to another based on example poses. However, stylizing the animations remains manual and requires the user to have a good understanding of the animation principles in these works. Motion Amplifiers is perhaps the work most similar to ours

for 2D stylized animations [33] (differences from our work elaborated in Appendix A). It provides preset motion effects (*amplifiers*) based on the animation principles. The user animates the position and rotation of a static sketch via direct manipulation and adds *amplifiers* to stylize the original animations. Monster Mash [18] and Wonder Painter [14] streamline the 3D animation pipeline [10] by inflating 2D sketches into 3D models. Animations are either created by moving control points of the inflated model [18] or automatically generated with limited customization options [14].

Several challenges in designing authoring interfaces need to be considered. The first is the balance between ease of use (understandability and usability) and expressiveness (diversity of results) [32]. Many tools mentioned above lean toward one side of this balance. The commercial tools are optimized for expressiveness and therefore hard to learn and use [48]. Tools that simplify existing workflows [18, 28, 58] are easy to use but are limited in terms of the stylized results. Secondly, the lack of modularity in the interface can limit either ease of use or expressiveness [33]. Tools that use keyframes [7, 8] or control points [18, 28] require the user to manually define what happens at each time step. This representation forces the user to break down a stylized effect into atomic components, making it difficult to edit as a whole afterwards. For expressiveness, the lack of modularity can make it difficult to composite multiple stylizations to create more diverse behaviors [18, 28]. The space-time curves in Space-Time Sketching [22] are modular and can be composited with each other for complex behaviors. Although Motion Amplifiers supports modularity via *amplifiers*, its inability to edit them limited its expressiveness [33].

In our work, instead of creating a fully-fledged animation tool, we focus on the relatively less explored area of 3D stylized animation authoring for users who might not be professional animators. One of our primary goals is to address the balance between ease of use and expressiveness. As discussed by Truong et al. [57], cognitive psychology research found that people mentally segment procedural tasks into coarse-grained events focusing on objects and fine-grained ones focusing on actions. We thus employ a layered authoring interface in which the surface layer (timeline sequencer) is optimized for ease of use and the second layer (node graph) provides more control to create custom stylized effects.

2.2 Authoring via visual programming

Visual programming often uses a node graph representation [25] and is a popular content authoring paradigm [25, 30, 61] because it enables non-programmers to produce complex and expressive results [42]. Prior work explored authoring via visual programming in 2D and immersive domains. In Dynamic Brushes [25], stylized brush strokes are created by mapping stylus input to procedurally generated patterns via a node graph. Kitty [30] enables the user to define visual, spatial, and other relationships between elements of an illustration for interactive behaviors via a relational graph. In Object-Oriented Drawing [60], attributes of a visual element like shape and stroke are abstracted as nodes that can be linked to other elements. Changes to a node are reflected in all linked elements to allow for consistent appearance and fluent mass editing. FlowMatic [61] allows the user to create interactive applications by specifying how objects should react to events like collisions and user inputs via a node graph in virtual reality.

Overall, using node graph representations for stylized animations remain relatively less explored. Some commercial tools [17, 51] use visual programming for motion or visual effects but take significant effort to master. We are motivated by the above works to adopt a node-graph authoring paradigm as our second level UI for greater expressiveness of our system in an intuitive manner.

2.3 Physics in 3D animation

Handling contacts and collisions is fundamental to physically reasonable effects in 3D animation [41]. Collision handling consists of two stages: *detecting* a collision and *responding* to it [41]. Physics-based animation in general addresses both through simulations [9]. However, this approach can be hard to direct [11] and inhibits animators' creativity and artistic expression [20]. Controls over the outcome of the final animations are limited to initial parameters [29], and a significant amount of tuning is often required to produce desired results [28]. To improve on this, recent work separates an object's motion control from its simulated physical reactions [16, 28]. Coros et al. proposed a method that allows the artist to control the movement of an object and define example shapes the object can deform to [16]. The models are then deformed according to the examples in reaction to contacts and collisions.

While parts of the 12 principles of animation reflect physical reality like follow through and squash and stretch [34, 55], the principles overall encourage animators to use exaggerated motions to convey interesting stories [20], which Barzel et al. [9] summarized as "visually plausible motion". Our system focuses on this type of effects and separates basic animations like translations from stylizations. First, event markers are put on the timeline when collisions are *detected* (Figure 1c). With ample amount of control, the user can make objects *react* to collisions by adding stylizations around the marked time and tuning them using collision information.

3 DESIGN STUDY

We conducted a design study to understand existing authoring paradigms of stylized 3D animations to inform our system design.

3.1 Methodology

A mixed-method approach is used in the study, consisting of an analysis of tutorial articles and videos and interviews with 6 professionals in the 3D animation industry. We first collected and analyzed tutorials of state-of-the-art 3D animation software including Maya, Blender, Houdini, Aero [2], After Effects [3], Cinema 4D, and KeyShot [38]. We studied each tool's authoring mechanism and how it would be used to create the stylizations from the 12 principles. To gain more in-depth insights and provide validation to our findings, we interviewed 6 professionals (2 animators *A1* and *A3*, 2 animation technical directors *A2* and *A4*, 2 design tool managers *A5* and *A6*). Each interview lasted 30 to 60 minutes. We asked them to speak from their experience about the pros and cons of existing 3D animation software and how our tool should be designed with novices in mind, and solicited their opinions on how each of the 12 principles of animation should be applied to 3D animation.

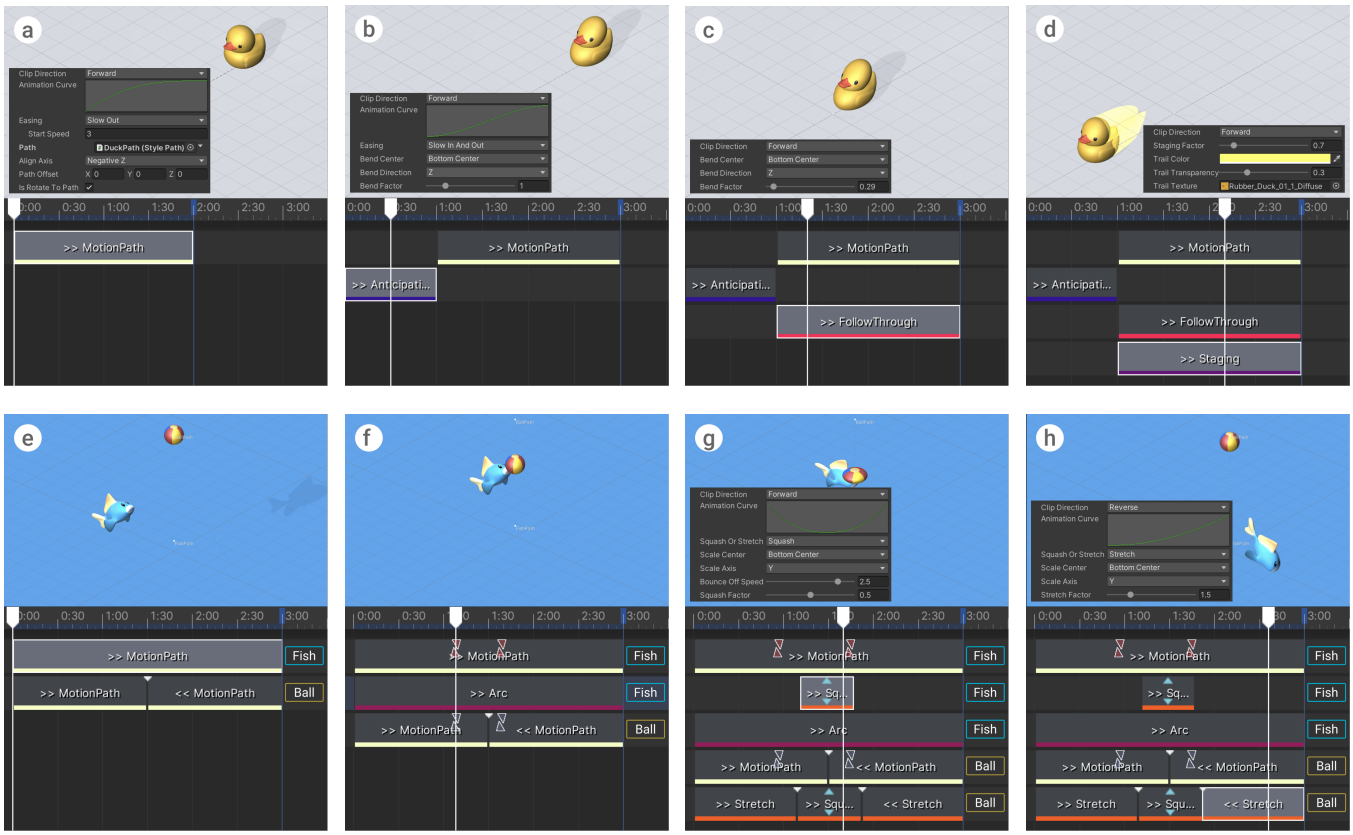


Figure 2: *Example stylized animation creation workflows.* Please refer to the supplementary video for live actions. (a): A *motion path* clip is added to move the duck along the 3D spline. (b): *Motion path*'s *Easing* is changed to Slow-out so the duck moves faster at the start. An *anticipation* clip that prepares the duck for dashing forward is added. (c): A *follow through* clip is added for a sense of speed. (d): A *staging* clip is added to attract the viewer's attention. (e): The ball and the fish are moved by *motion path* clips similar to (a). (f): An *arc* clip is added to make the fish more lively. (g): *Squash and stretch* clips are added based on the event markers. (h): The user moves the motion path and adjusts *squash* clips' parameters to fine-tune collision reactions.

3.2 Existing tools and paradigms

We adopted the categorization of animation authoring paradigms by Thompson et al. [56]: keyframe-based, procedural-based, and preset-based, and discuss each category's advantages and drawbacks.

Keyframe-based. This paradigm can be found in many popular animation software [7, 21, 39]. The user inserts keyframes at specific positions in a timeline to control how attributes of a 3D object (e.g. rotation, scale) should change over time. Although versatile and used by many professionals, this approach can be hard to learn for beginners (*A4* and *A6*). A6 further suggested that keyframing coarse-grained motions is simple, but creating stylized effects can be challenging. For example, a typical bouncing ball animation with squash and stretch requires the user to key 9 different attributes of a sphere: `Rotate XYZ`, `Scale XYZ`, and `Position XYZ`. These keyframes need to be positioned temporally in coordination with each other in order to look convincing. Editing the squash and stretch effect is also difficult because each keyframe needs to be adjusted individually.

Procedural-based. In this approach, users create animations via visual programming [56]. A popular representation is the node graph, which is employed by many programs we researched [2, 17, 51]. This paradigm enables non-programmers to create expressive visual results [42]. However, A4 pointed out that "a node graph filled with nodes can seem intimidating to novice users." Sousa et al. also stated that large graphs can cause visual cluttering [52]. Moreover, the lack of a timeline-like metaphor makes it difficult to coordinate the timing of different animations [33], limiting the expressiveness of this paradigm.

Preset-based. Animating with presets has the benefits of a lower learning threshold and reduced time and effort [56]. Design tools like PowerPoint [40] and XD [4] provide template animation and transition effects. Motion Amplifiers [33] offers preset effects for 2D sketches, and Aero has an animation library [2]. However, this paradigm also suffers from the lack of a timeline in terms of expressiveness [33] and ease of use: "Users of Aero have been saying that timing different animations is a pain" (*A5*).

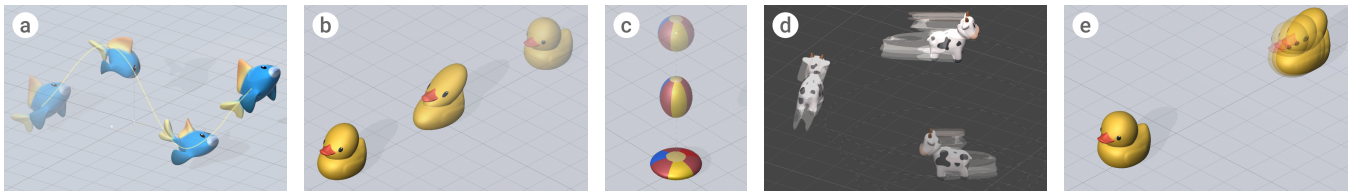


Figure 3: *Example stylizations in our preset library:* (a) arc, (b) follow through, (c) squash and stretch, (d) staging, (e) anticipation.

Hybrid. It is common for animation tools to employ multiple authoring paradigms [56]. After Effects has preset effects for layers in a timeline, and Blender offers a node-based plugin [37] in addition to its timeline. Hybrid approaches in these tools are designed for speeding up an existing workflow or offering alternative ways for similar outcomes. Example-based approaches automatically interpolate between example states, and state transitions are triggered by factors like time [8] and events [28]. Sketch-based tools allow for animating via direct manipulation, and stylizations can be added via presets [33] or more sketching [22]. Overall, hybrid authoring interfaces combine the strengths of various authoring paradigms to offset their individual flaws.

3.3 Insights and discussion

Various animation authoring paradigms exhibit different strengths in terms of ease of use and expressiveness. While keyframe and procedural-based software gives the user full control, it requires a significant amount of learning and the user often needs to create everything from scratch. On the other hand, animating with presets can speed up the workflow and has a lower barrier of entry but can be limited in terms of expressiveness. The balance between ease of use and expressiveness is summarized by A6 as the *Curation* → *Customization* → *Creation* spectrum. *Curation* is montaging presets from a library like curating an exhibition (ease of use). On the other hand, *Creation* means building everything from scratch with full control, like keyframing a bouncing ball animation (expressiveness). *Customization*, situated in the middle, gives the user a base to build upon with the ability to make further changes.

This spectrum converts the ease of use-expressiveness trade-off into a sliding scale that encourages the combination of authoring paradigms in a **layered** manner to support different levels of control that a user may need depending on their experience and target outputs. This idea was supported by Shneiderman et al. [50] and echoed by A2, who suggested to us to design the system not only with intuitive presets for users near the *Curation* end but also with advanced controls to move them through the spectrum as they become more experienced. Moreover, while some effects like follow through have defined visual qualities, others like anticipation vary from context to context [34]. A layered approach thus allows us to define default behaviors for vaguely defined stylizations while enabling them to be modified based on contexts.

3.4 Design goals

Based on our design study, we have formulated a set of design goals for our stylized 3D animation authoring system.

Layered authoring interface for ease of use and expressiveness. Our primary goal is to fluidly balance between ease of use and expressiveness. To achieve this, we should employ a layered authoring paradigm because of its flexbility and its ability to mitigate the drawbacks of individual paradigms as discussed above. The user should be able to choose which authoring paradigm they want to use based on their familiarity with the tool and desired results.

On the surface level, we should employ a timeline sequencer but refrain from incorporating keyframing to make it intuitive to use and easy to learn (*A4* and *A6*). To harness the strength of the preset-based authoring paradigm, the user should be able to directly add and position predefined stylizations to the timeline for easy coordination of different animations. Once the user becomes comfortable with the surface level UI, they should be able to further customize or define stylizations through a second level authoring interface [50]. Since visual programming has the benefit of increasing the expressiveness of a system while being relatively easy to use, we adopt a node graph representation as the second level UI while keeping it straightforward to edit (*A4* and *A6*).

Efficiency and modularity. Our system should make the process of authoring stylized animations efficient by providing a library of preset stylizations based on the 12 principles of animation (*A4*). Each preset should provide a set of adjustable parameters for fine-tuning and should be modular for ease of editing and composability with other stylizations for more complex effects (*A6*). The benefits of modularity on expressiveness of the authoring system has been demonstrated by works discussed above [22, 33].

Real-time visually plausible physics. Our tool should focus on stylizing animations with *visually plausible motions* [9]. The authoring of original animations and stylizations should be separated for ample control. Collisions should be automatically detected and marked to assist the user with timing animations in reaction to collisions with real-time visualizations and feedback (*A2* and *A4*).

4 USER INTERFACE AND INTERACTION

Based on our design study and goals, we have implemented a stylized 3D animation authoring system that consists of a layered user interface and a library of preset stylizations. The term "stylization" in our system refers to a type of animation, which can be either plain object transformations like translation and rotation, or motion effects based on the 12 principles of animation, like squash and stretch and follow through. We describe these components below, and illustrate how they work together in example authoring sessions shown in Figure 2.

4.1 Layered authoring interface

Our layered authoring interface is composed of two levels. On the surface level, the user works with a **timeline sequencer** that contains tracks and clips to compose stylized animations in a modular manner (Figure 1c). Beneath the surface level, a **node graph** defines the behavior of each preset stylization (Figure 1d). Once the user becomes familiar with the timeline sequencer, they can work with the node graph to create custom stylizations (Figure 8), which extend the preset library beyond the 12 principles of animation. Note that users new to our system are able to only work with the timeline sequencer without editing the node graph.

Timeline sequencer. The timeline sequencer looks similar to the conventional timeline tool found in popular 3D animation software like Maya and Blender (Figure 1c). However, instead of storing keyframes that control various object attributes, our timeline sequencer consists of clips grouped by tracks (rows). Each clip represents a modular unit of animation, and all clips in the same track have the same type of stylization. An object binds to a track and is animated when the playhead of the timeline moves through a clip, whose length and position directly controls the duration and timing of animations. Multiple stylizations can be placed on multiple tracks bound with the same object to create more dynamic effects (Figures 2 and 8). For example, as shown in Figure 1b, while moving along a motion path, the fish is stylized with *arc*, *follow through*, *staging*, and *squash and stretch*.

Node graph. Each stylization in the timeline sequencer is internally represented by an automatically generated node graph (Figure 1d). Novice users are able to work only with the timeline sequencer to create stylized animations. As they become more familiar with our system, they have the option to create custom stylizations by editing the node graph (detailed in Section 5.3). The user can easily apply the same custom stylizations to multiple objects by assigning the same node graph to them. Overall, we keep the user's interactions with the node graph simple to ensure that people new to this paradigm can quickly learn to use it.

Event markers. To help users create animations that react to particular events like collisions, our system detects collisions between objects when the timeline is being played. Once a collision is detected, an event marker containing information about the collision is placed at the time when it takes place (Figure 1c). Collision information includes the name of the colliding object, collision point, and collision direction (normal). The user can use this information to adjust the parameters of the stylizations.

Motion path. Animating an object's position requires a 3D spline referred to as a motion path (Figure 3a), a common feature in many animation systems. Our system offers a freeform shape and 8 parametric shapes (star, circle, polygon, arc, ellipse, rectangle, helix, line) with adjustable parameters (radius, height, etc.). The user can add, remove, or translate vertices to define the 3D spline. All shapes have a ***Smoothness*** parameter that controls the curvature of line segments in between two vertices (0 makes it a straight line).

4.2 Preset stylizations

Our preset stylization library contains motion effects constructed from a subset of the 12 principles of animation and basic manipulators like motion path and rotation that are building blocks for other stylizations (Figure 3). Each preset has adjustable parameters that are exposed when a clip embodying it is selected (Figure 2).

Kazi et al. [33] classified the 12 principles of animation into 5 categories. Principles in the categories of *visual design* (solid drawing, appeal) and *the process of animation* (straight ahead and pose to pose) are not translatable to motion effects because the former applies to a different part of the 3D animation pipeline [10] while the latter pertains to animation drawing techniques. Within *geometric manipulation*, we also omitted secondary actions because our tool is intended to work with single-mesh solid objects.

4.2.1 Common parameters. A clip embodies a stylization and exposes its parameters (Figure 2). We first describe common parameters that apply to all stylizations. The first one is ***Clip Direction***, which controls the direction in which a clip is played. Changing it from Forward to Reverse plays the animation backward. The direction is indicated in a clip's name by "»" or "«" (Figure 1c). The second one is ***Animation Curve***, which visualizes the underlying mapping from a clip's local time to an output `progress`. For preset stylizations, the curve reflects changes made to the ***Easing*** parameter (Figure 2a). For custom stylizations, it displays the animation curve defined in the node graph.

4.2.2 Basic manipulations. These stylizations are basic manipulators common in many animation systems.

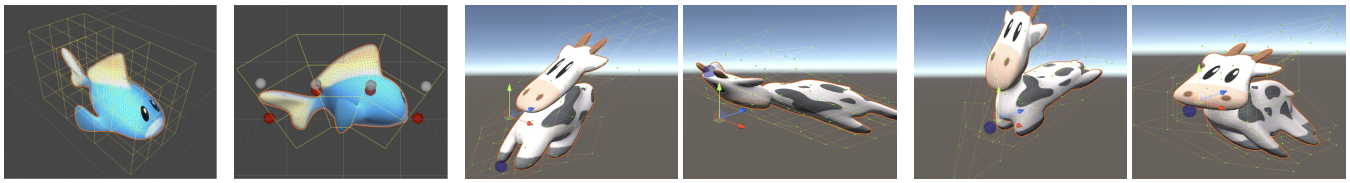
Motion path. This stylization moves an object over time to travel along the 3D spline assigned to the ***Path*** parameter (Figure 2a). The user can offset the object from the path by changing ***Path Offset***. ***Is Rotate To Path*** controls whether the object's ***Align Axis*** should be constantly rotated to the direction of the motion path's tangent.

Scale. This effect changes an object's scale from ***Start Scale*** to ***End Scale*** over the duration of the clip. ***Scale Center*** defines the origin from which the object is scaled, and comes with 3 options - Top Center, Center, and Bottom Center - to help novice users more easily adjust the origin in object space. These options are chosen due to their common usage and because the *squash and stretch* stylization requires scaling from one end of the object (e.g. ball hitting the floor). Users who understand object space coordinates can use the Freeform option to freely define ***Scale Center***.

Rotate. Rotate animates the rotation of an object from ***Start Rotation*** to ***End Rotation*** over the duration of the clip.

Bend. This effect deforms the object in a "bending" manner from ***Bend Center*** along ***Bend Direction*** (Figure 4). ***Bend Factor*** controls the amount of bending (Figure 2c). Similar to ***Scale Center***, a Freeform and common options are provided for both ***Bend Center*** and ***Bend Direction***. ***Bend Center*** have the same options as ***Scale Center***, and the common options of ***Bend Direction*** are the 3 axes of an object in positive and negative directions.

4.2.3 Stylizations from principles. Here we describe the subset of the 12 principles of animation in our preset library.



(a) Free form deform (b) Arc deform (c) Bend deformation via origin (d) Bend deformation via direction

Figure 4: *Mesh deformations via free form deformer (FFD).* Every model is initialized with a $4 \times 4 \times 4$ FFD shown in (a). In (b), the gray anchor points of the *Align Axis* of the FFD are translated to corresponding red points on the motion path, while the rest of the vertices of the FFD follow passively. Bend deformations can be controlled by changing *Bend Origin* and *Bend Direction* parameters. *Bend Origin* is Bottom Center and Top Center in (c), and *Bend Direction* points up (Y) and right (X) in (d).

Slow-in and slow-out. This affects how other stylizations are played via the ***Easing*** parameter (Figure 2a) with 4 options: Slow-in & Slow-out, Slow-in, Slow-out, and None. Slow-in slows down the animation at the beginning, while Slow-out at the end. The default Slow-in & Slow-out does both (Figure 5a). Selecting Slow-in or Slow-out exposes ***End Speed*** or ***Start Speed*** that controls the animation speed. Both speed options are exposed when None is selected (Figure 5b). These options allow for easy control of animations without the need for manual adjustments of value curves in the traditional keyframing approach. For example, the Slow-out option with a high ***Start Speed*** can be assigned to a *motion path* clip that follows an *anticipation* clip to convey the idea that the *anticipation* action makes the object move faster initially (Figure 2).

Squash and stretch. This stylization stretches or squashes the object along the ***Scale Axis*** from ***Scale Center*** (Figure 3c). The ***Scale Axis*** can be set to one of the X, Y, Z axis. A full squash and stretch motion consists of three parts - stretching out from an initial shape, squashing down when stopped, and stretching back to the initial shape. The effect is thus divided into two sub-actions: a *stretch* action and a *squash* action, which can be selected through the ***Squash or Stretch*** parameter (Figures 2d and 2e). The amount of stretching or squashing is controlled by ***Stretch Factor*** or ***Squash Factor*** respectively.

Follow through. This effect mimics how some parts of an object lag behind other parts near the source of motion due to inertia (Figure 3b). It bends the object from ***Bend Center*** along ***Bend Direction*** when the object is moving. By default, the ***Bend Center*** is set to Bottom Center as the source of motion usually comes from where the object touches the ground (Figure 2c).

Arc. The arc principle has two interpretations, both of which are supported. The first means that an object should move in curved paths instead of straight lines [55] and is controlled by the ***Smoothness*** parameter of the 3D spline motion path described above. The second is deforming the object along its motion path [22, 33] and achieved through the *arc* stylization type (Figure 3a).

Staging. The original staging principle is a broad concept about leading the audience's eye to "exactly where it needs to at the right moment" [34]. We took inspiration from prior work [33, 47] and added a motion trail behind the object to draw the viewer's attention (Figure 3d). ***Trail Color***, ***Trail Transparency***, and ***Trail Texture*** are used to adjust the appearance of the motion trail, while ***Staging Factor*** controls its length (Figure 2d).

Anticipation. Anticipation is another broadly defined animation principle meaning "the preparation for an action" [34]. The ambiguity of this principle is echoed by *A6* in our design study: "Anticipation is tricky because you don't know what it looks like." To address this, we play to the flexibility of the node graph by allowing the default behavior of the *anticipation* stylization to be customized.

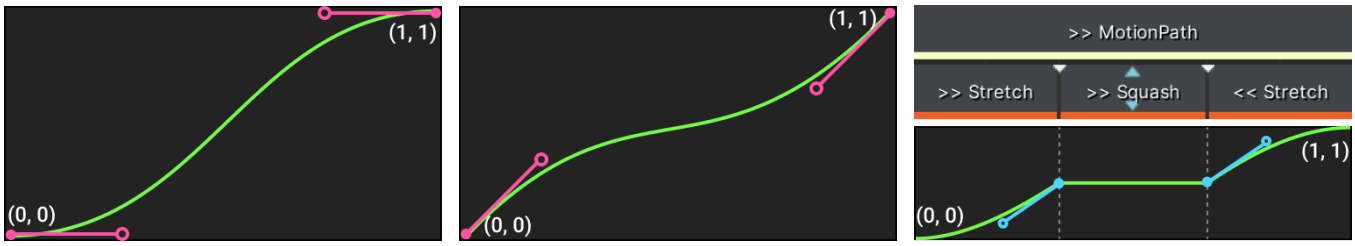
By default, *anticipation* gradually bends an object backward like a stationary *follow through* (Figure 3e). This default behavior is inspired by A1 in the design study, who described the anticipation behavior as "pulling on a spring, holding it there for a bit before releasing it." The user can use the node graph to change the default bending to either scaling or rotating as these manipulations do not require a 3D spline to be drawn. Example use cases include a balloon scaling up before loosing air and a product turntable animation before the exploded view. Broadly, our timeline sequencer extends *anticipation* to any stylization placed immediately before another motion, based on the anticipation principle's original definition [34].

4.2.4 Implicitly supported animation principles. In addition to the stylizations described above, our system enables implicit applications of the *Timing* and *Exaggeration* principles. *Timing* is defined as "the amount of time that action will take on screen" [55]. The user can adjust the speed of an animation either through the length of a clip or the ***Easing*** parameter. *Exaggeration* is embedded in the adjustable parameters of each stylization. For example, one can exaggerate *squash and stretch* or the *follow through* inertial force by increasing corresponding factors.

5 IMPLEMENTATION

Our system is developed in Unity [53] with C# under macOS 11.4 Big Sur. The timeline sequencer is built with Unity's Timeline API [54], and the node graph with xNode [13]. The 3D splines and the free form deformers (FFD) [49] are part of MegaFiers [59]. A $4 \times 4 \times 4$ FFD is created for a given 3D model with the same size as its bounding box (Figure 4a). Translating the FFD's vertices deforms the object mesh, and different deformations are composited together by combining each vertex's translation vectors.

Unity's physics engine detects collisions between object colliders every frame. To ensure that our system can always produce real-time previews, we use a collider with the same shape as the object's



(a) Default animation curve (b) Adjusted animation curve (c) Animation curve with auto-pause

Figure 5: *Manipulating animation curves through preset stylizations.* (a) is the default animation curve with *slow-in and slow-out*. (b) is the result of setting *Start Speed* and *End Speed* to 1. In (c), a flat region is added to the animation curve of the *motion path* clip corresponding to the *squash* clip. The blue handles are controlled by the *Bounce Off Speed* parameter.

FFD, instead of one that has the same shape as the object's mesh, which can produce more accurate detection results but becomes computationally expensive as the mesh increases in complexity.

A clip represents a modular unit of animation with a specific stylization type. The behavior of that stylization is defined and executed by its linked node cluster in the node graph (Figure 6). Adjustable parameters associated with that stylization are stored within each clip. When a clip is played, it passes these parameters to the linked node cluster, which takes them as input to execute the animation. This allows the user to set different parameter values for different clips of the same stylization type (Figure 2).

5.1 Graph nodes

A node cluster consists of three nodes: Animation, Action, and Effect (Figure 1d).

5.1.1 Animation node. This node is a time warper via a Bézier animation curve. It maps the input time `t` (0 to 1 in a clip's local time) to an output `progress` with two end points anchored at (0, 0) and (1, 1). The slope coming into a point, `inTangent`, and the slope coming out a point, `outTangent`, of the two end points are controlled by the ***Easing*** parameter in preset stylizations (Figures 5a and 5b). The animation curve can be directly edited in custom stylizations.

5.1.2 Effect node. This node links the node cluster to a particular stylization through its ***Stylization Type*** parameter and has no output. To complete a node cluster, both an Animation node and a type of Action node need to be connected to an Effect node. When a clip is played, it passes information including parameters and local time to the Effect node, which then calls on the connected nodes to animate objects. For example, to perform *slow-in and slow-out*, the Effect node updates the tangents of the animation curve in the Animation node. For *squash and stretch*, the Effect node triggers volume-preserving scaling in the Scale Action node.

5.1.3 Action nodes. These nodes execute animations based on the `progress` input from the Animation node and stylization parameters. There are 6 types of Action nodes in 2 categories: *Basic transformations* and *Mesh and shader manipulations*.

– *Basic transformations* include:

Motion Path This node moves an object along a 3D spline by mapping `progress` to a point on the spline.

Rotate This rotates an object by linearly interpolating between ***Start Rotation*** and ***End Rotation*** by the interpolant `progress`.

Scale This node animates the scale of an object from ***Start Scale*** to ***End Scale*** with the origin at ***Scale Center***. Each FFD vertex is translated by the position vector of ***Scale Center***, multiplied by a scaling matrix, and then translated back.

– *Mesh and shader manipulations* include:

Arc Deformation This node deforms the object along a motion path at `progress` (Figure 4b). Anchor points along an object's ***Align Axis*** (gray) are projected to corresponding points on the path (red). The vertices of the FFD are then rotated accordingly to deform the object.

Bend Deformation This node creates a vector $\vec{V}_B$ from ***Bend Origin*** along ***Bend Direction*** and computes the distance D_i between each FFD vertex and $\vec{V}_B$ (magnitude of the vector between a vertex and its projection onto $\vec{V}_B$). Each vertex is then translated by $\vec{V}_B \cdot D_i \cdot$ ***Bend Factor*** (Figure 4).

Motion Trail This node updates the `Direction` vector of a custom motion trail shader written in HLSL to draw the trail in the opposite direction of the object's movement (Figure 3d). The custom vertex shader scales down the object mesh slightly and extrudes each mesh vertex along `Direction` by the amount of ***Staging Factor*** with added perlin noise.

5.2 Building preset stylizations from nodes

Motion path, Scale, Rotate, and Bend. These 4 presets use their corresponding Action nodes to animate objects.

Slow-in and slow-out. This stylization changes the `inTangent` and `outTangent` of the points of an animation curve to achieve the easing effects. The ***Start Speed*** and ***End Speed*** parameters assigned through a clip are respectively set to the `outTangent` of the start point and the `inTangent` of the end point (Figure 5a).

Squash and stretch. This stylization uses a Scale node to scale the object in a volume-preserving manner (Figure 3c). It scales the dimension corresponding to ***Scale Axis*** by `ScaleFactor` and the other two dimensions by $\frac{1}{\sqrt{\texttt{ScaleFactor}}}$. `ScaleFactor` is calculated by evaluating the animation curve at clip local time t. The animation curves of *stretch* clip starts at (0, 1) and goes up to (1, ***Stretch Factor***), while the animation curve of a *squash* clip starts

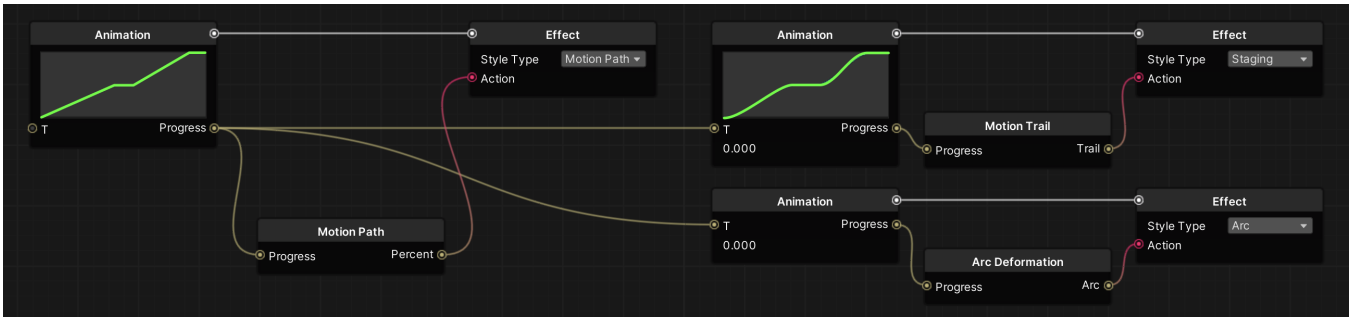


Figure 6: *Node clusters of 3 preset stylizations: motion path, arc, and staging. Arc* and *staging* takes input from *motion path*'s Animation node to ensure proper behavior. *Staging*'s smoothed curve is constructed by setting all tangents of points to 0.

at (0, ***Stretch Factor*** of the preceding *stretch* clip), goes down to (0.5, ***Squash Factor***), and then ends at (1, ***Stretch Factor*** of the following *stretch* clip) as shown in Figures 2g and 2h. This ensures smooth transitions between each clip.

Since the squashing action requires the object to stop moving briefly in a collision event, pauses are automatically added to the *motion path* clip when a *squash* clip overlaps with it on the timeline. A pause is a flat line segment with two end points, and multiple *squash* clips adjacent to each other create one consecutive pause (Figure 5c). The ***Bounce Off Speed*** parameter of the *squash* clip controls the speed at which the object comes into and out of the collision by changing the `inTangent` of the point at the start of the pause and the `outTangent` of the point at the end of the pause (blue handles in Figure 5c).

Follow through. It uses the Bend Deformation node and adjusts the bending amount according to the object speed, in addition to the ***Bend Factor*** parameter. No *follow through* takes place when the object is stationary (Figure 3b).

Arc. Arc uses the Arc Deformation node and requires a *motion path* clip to be present to deform the object along the path (Figure 3a). The time `t` value of *arc*'s Animation node takes input from that of *motion path* to ensure proper behavior (Figure 6).

Staging. Staging uses the Motion Trail node to draw a trail behind the object (Figure 3d). The Animation node of *staging* also takes input from that of *motion path* and recreates a smoothed version of *motion path*'s animation curve to ensure that the motion trail does not change in length abruptly (Figure 6). The smoothening is achieved by setting all point tangents to 0, and the trail length is calculated as ***Staging Factor*** $\times$ *Slope of the smoothed curve at* `t`.

Anticipation. By default, *anticipation* uses a Bend Deformation node (Figure 3e). Its behavior changes accordingly when the Action node is changed to either Scale or Rotate (Figure 8d).

5.3 Custom stylizations

To create a custom stylization, the user builds a new node cluster in 3 simple steps (see Figure 8 for examples): (1) Choose an Action node. (2) Set the Effect node's ***Style Type*** to Custom. (3) Edit the animation curve in the Animation node to define desired behaviors.

6 EVALUATION

We conducted a user evaluation with both professional and novice animators to investigate the following research questions: (1) How does our system compare with the traditional animation software in terms of ease of use, time to completion, satisfaction, and expressiveness? (2) How does our layered authoring interface address the balance between ease of use and expressiveness (i.e. are the participants able to work with the node graph after becoming familiar with the timeline sequencer)? Inspired by prior creativity support tools that were also designed to help users with content authoring [33, 45, 46, 61], we have designed our user study as follows.

6.1 Participants

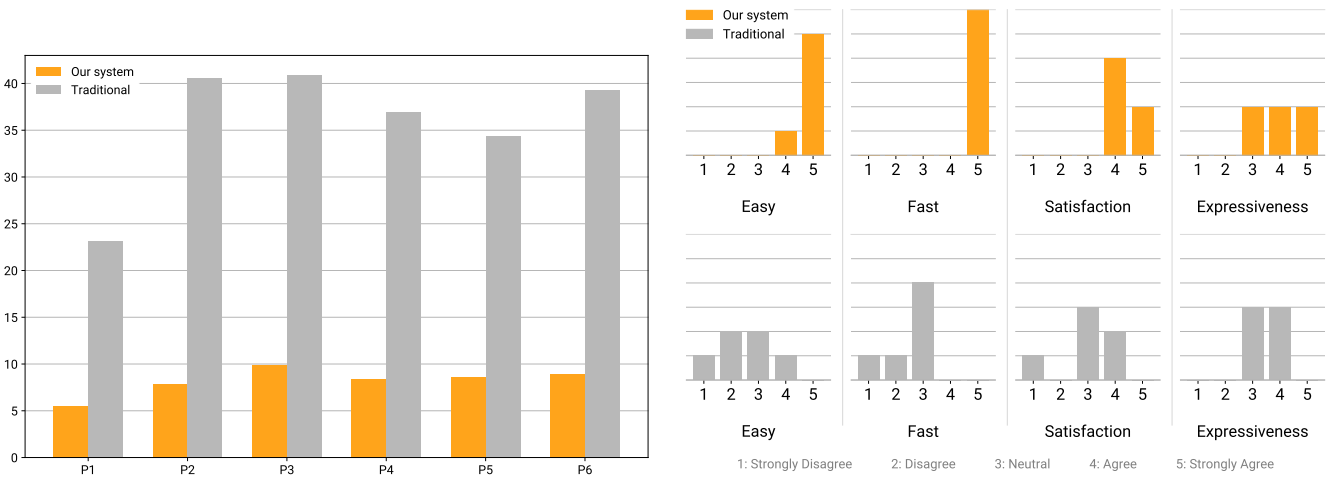
We recruited 6 participants (2 female, 2 male, 1 non-binary, 1 prefer not to say) to evaluate our system. 2 of them were professionals (*P1, P2*) and 4 were novices (*P3, P4, P5, P6*). P1 has 1 year of experience, P2 has 4 years, and both use Cinema 4D in their professional practices. The novice participants have experience with Blender (ranging from beginner to intermediate based on the post-study questionnaire). Each participant was compensated for their time.

6.2 Procedure

All the experiments were conducted on a 15-inch laptop. Each study was approximately 2 hours and consisted of 4 sessions: *training*, *target*, *open creation*, and *interview and feedback*. The participants were encouraged to think aloud during the sessions. The instructor (first author) was present to provide guidance, observed participant behaviors, and collected animation data and feedback.

Training (30 minutes). Each participant first watched a short video to get an overview of our system. After introducing the participant to the 12 principles of animation, the instructor walked them through a step-by-step tutorial of the system. The participant was then given 10 minutes to freely explore the system.

Target (60 minutes). The participants were asked to reproduce the two target stylized animations shown in Figure 2 in two conditions: a baseline condition with a traditional 3D animation software and a condition with our system. The two target animations together employed all non-basic stylizations in our preset library. In the baseline condition, the participants were free to choose which software they wanted to use. The 2 professional participants (*P1,*



(a) Target session completion time (minutes)

(b) Subjective results from post-study questionnaire

Figure 7: *User study results.* Since only our tool was used in the open session, the expressiveness ratings of the traditional tools were reported by the participants based on their prior experiences.

P2) used Cinema 4D and the rest used Blender. Both tools are primarily keyframe-based but also have node-based and preset-based functionalities. The same models and motion paths were provided to all participants in both conditions. The condition orders were fully counterbalanced among participants, and there was no time limit for each condition.

Open creation. The goal of this session is to evaluate the expressiveness of our system as well as identify strengths and issues. Each participant was given a set of 3D models to use and was free to explore the tool to create stylized animations. They were asked to draw motion paths if needed, and were encouraged to work with the node graph to change the behaviors of the *anticipation* stylization or create custom ones.

Interview and feedback. The participants were asked to fill out a questionnaire to rate various aspects of our system and record their background information. The instructor then interviewed each participant for feedback.

7 RESULTS

Our user study results are summarized in Figure 7. All participants completed the target session in both conditions and in the baseline condition all authored animations primarily using keyframes.

7.1 Quantitative measurement

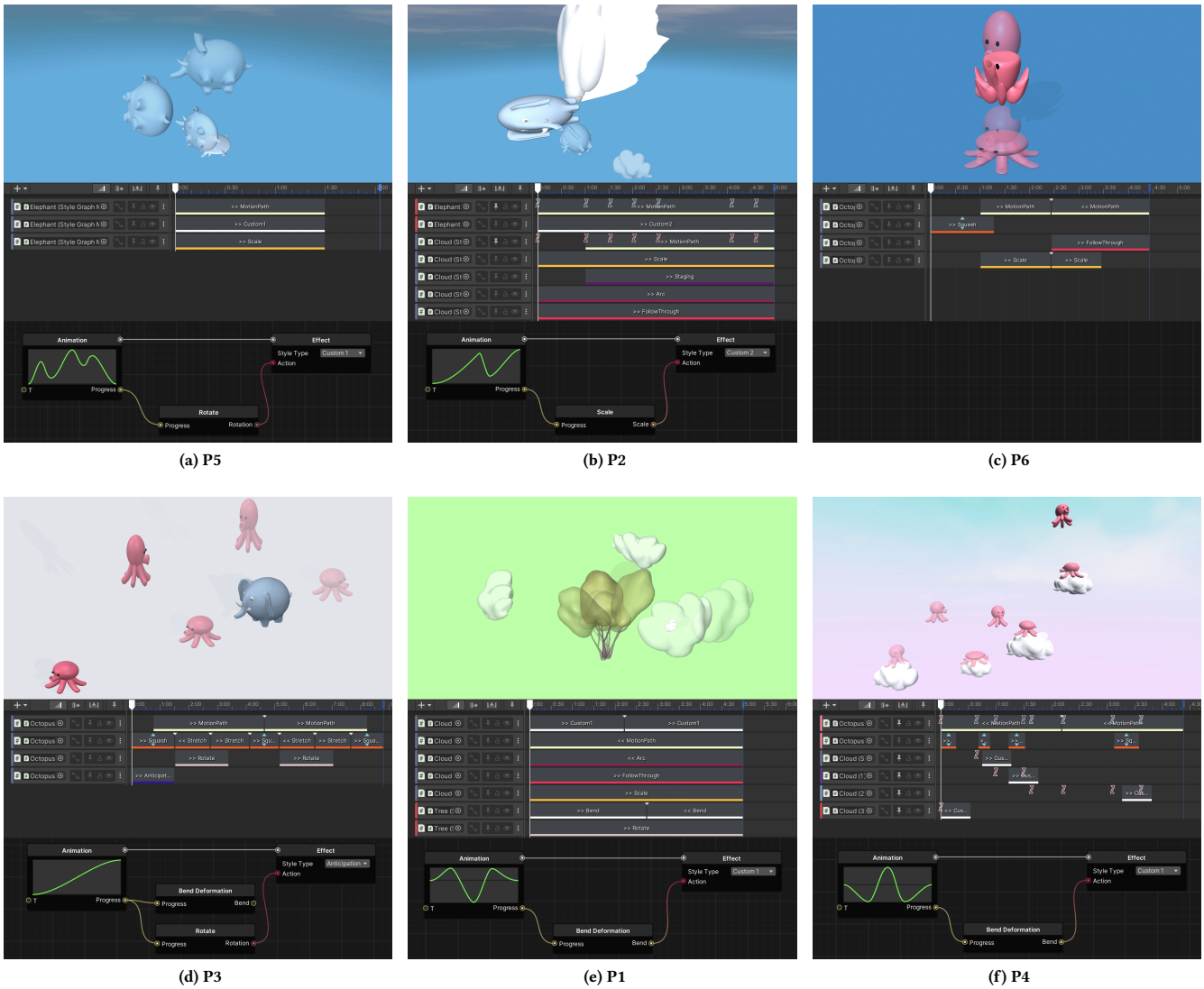
On average (Figure 7a), the target session took 36 minutes (sd = 6.72) for the baseline condition and 8 minutes for our system (sd = 1.45). The effect of condition on completion time was significant ($t(5) = 9.87$, $p < .001$). This indicates that our system is more efficient and faster to use.

In the baseline condition, novice users (*P3, P4, P5, P6*) struggled with using mesh deformers to achieve the bending effects of *anticipation* and *follow through* (first animation) and scaling the ball from its bottom for *squash and stretch* (second animation). P5 and P6 resorted to online help during their sessions. P2 reproduced the *staging* motion trail with a transparent material but needed to work with Cinema 4D's renderer which does not update in real-time. These struggles accounted for longer completion time. With our system, P5 and P6 mistakenly used *anticipation* instead of *follow through* but recovered independently after rewatching the first animation. P1 and P4 missed the subtle *arc* deformation in the second animation but were able to add it when reminded.

7.2 Subjective feedback

In the post-study questionnaire, the participants compared the two conditions based on a set of 1-5 Likert scale questions (higher is better) as shown in Figure 7b. Overall, the participants found our system to be easier and faster to use and were more satisfied with the experience than the baseline condition. They also found our system to be somewhat more expressive than the traditional tools.

7.2.1 System. The participants in general found our system to be beginner friendly. For the timeline sequencer, the participants thought that it is an intuitive metaphor (*P4, P5*) and easy to use (*P2, P3, P5*). When working on the target animations in the baseline condition, P2 mentioned that they have to plan what should happen at each frame in order to build the animation. In contrast, with our system, P2 "just needs to coordinate different clips" and found this way to be "more straightforward than keyframing." P1 and P3 liked how clips abstract motion effects as units and can be used out-of-the-box. In terms of composability, P3 commented that "the layering of clips is easy to visualize and allows me to compose more complicated effects." 4 participants found the node graph to be easy to work with after learning to use it with some guidance (*P2, P3, P5, P6*). Both P2 and P4 appreciated how it enables them to create custom stylizations and adjust the presets beyond the parameters provided. P1 liked how it enables the same effect to be



(a) P5 (b) P2 (c) P6

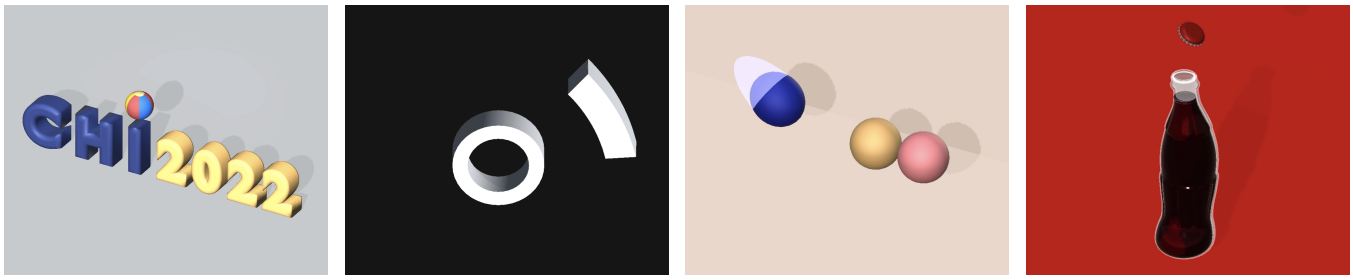
(d) P3 (e) P1 (f) P4

Figure 8: *Open session results.* Within each case are the sample animation frames (top), the corresponding timeline sequencer (middle), and the custom stylization in node graph (bottom). Section 7.3 provides detailed descriptions of these results.

applied to multiple objects (Figure 8f). P5 found that the customization capability to be "great for achieving organic effects that look more random" (Figure 8a). Overall, the feedback indicated that the *Efficiency and modularity* design goal has been achieved.

In terms of physics, P2 and P4 appreciated our system's real-time aspect. P4 and P6 liked how the tool enables them to animate models in a physical context: "I can combine different stylizations to create dynamics like pressure and gravity so the models move around with a sense of physical reality" (*P6*). 4 participants (*P2, P3, P4, P5*) mentioned that the event markers are useful when finding critical frames to position clips, such as timing squash and stretch in reaction to collisions (Figure 2g). P6 commented: "When there are many 3D objects, it is often hard to see what's happening no matter how you orient the viewport, so having the markers is really helpful for me to know when two things are going to touch each other." The feedback here supports the fulfillment of the *Real-time visually plausible physics* design goal.

7.2.2 Preset Stylizations. 4 participants (*P2, P3, P5, P6*) found the preset stylizations to be useful with intuitive parameters: "they are very encouraging to work with" (*P5*). P3 and P6 believed that users who are not familiar with the 12 principles of animation are still able to use the presets and consequently learn about the principles. 3 participants (*P2, P4, P5*) liked how the preset stylizations can speed up their workflow and serve as a source of inspiration for further creations, such as "I can quickly achieve something and use that as a base to work from" (*P5*) and "using the presets inspires me to create more because I can quickly iterate and prototype" (*P2*).



(a) Kinetic typography (b) Loading animation (c) Newton's cradle (d) Advertisement

Figure 9: *Example animations based on participants' suggestions.* Please refer to the supplementary video for live actions. (a) and (b) are example 3D motion graphics. (c) can teach kids about physics, and (d) is a soft drink commercial. These animations can be used in domains like education, marketing, and immersive AR experiences.

For specific stylizations, P1 and P5 found *arc* to be "neat, subtle, and satisfying." P6 thought that animating elongated models like dragons or snakes with *arc* could make them come alive, which is hard to do in traditional software. P4 and P5 found *slow-in and slow-out* to be convenient because in traditional tools they need to manually adjust keyframes. P5 mentioned that *slow-in and slow-out* "makes an animation looks more professional." P2 appreciated *staging* because it integrates a custom shader and visualizes in real-time: "I was struggling with reproducing it in Cinema 4D because I had to render the animation to see the transparent material and it was slow to make adjustments."

7.3 Sample results

Figure 8 shows animations by participants in the open creation sessions. 4 participants (*P1, P2, P4, P5*) created custom stylizations and one (*P3*) changed the *anticipation* behavior via the node graph.

P5 and P2 created custom stylizations to add organic sense and randomness. P5 animated an elephant balloon loosing air (Figure 8a). The Rotate node is controlled by a sine-wave looking curve to mimic how the balloon flies randomly through the air. In P2's animation, the elephant and cloud move up spirally in a tornado (Figure 8b). Their custom stylization is made with the Scale node to organically change the elephant's size as if being stretched by the tornado.

P6 and P3 brought out the soft-bodied nature of the octopus model. In P6's animation (Figure 8c), they timed the slow falling of the octopus with an upward bending *follow through* to animate the soft tentacles lagging behind due to air resistance. In P3's circus show (Figure 8d), they changed *anticipation* from bending to rotation. The octopus thus rotates around to "greet the audience" while preparing for the jump. P3 further used *squash and stretch* in every jump with high factor values to convey its soft-body nature.

P1 and P4 bend the cloud model in custom ways. In P1's animation (Figure 8e), a "wing flapping" effect is created with the Bend Deformation node so that the cloud moves lively around the tree. P4 made a platform-game-style animation in which the octopus jumps from one cloud to another (Figure 8f). This is a challenging task since multiple collisions took place between objects. They created a "shake" effect with the Bend Deformation node for the clouds in reaction to the octopus' landing. All the clouds were connected to the same node graph to share the same "shake" stylization.

7.4 Example animations

As shown in Figure 9, we created example animations to demonstrate the expressiveness of our system based on participants' answers to the question "What kind of animations do you think this system is suitable for creating?" The participants thought our tool is ideal for authoring 3D motion graphics, like kinetic typography (*P2*), loading animation (*P1*), and animated logos (*P3*). In terms of application domains, 4 participants (*P2, P3, P5, P6*) mentioned advertisements that showcase physical products, two mentioned educational contents for kids (*P3, P6*), and one mentioned interactive animations for augmented reality (AR) experiences (*P1*).

8 DISCUSSION

As demonstrated above, the last two design goals and the first research question have been supported. The first design goal and the second research question both focus on how the balance between ease of use and expressiveness is addressed through the layered authoring interface paradigm. The subjective feedback and ratings (Figure 7b) have shown that our tool is easy to use for both beginners and more advanced animators. We are delighted to find that the participants also rated the expressiveness of our system to be somewhat higher than the traditional tools (Figure 7b). P4's comment partially reveals the reason: "The traditional tools are very powerful, but it takes a lot more effort to fiddle with."

The layered authoring interface paradigm requires gaining familiarity from the users in order to move from the surface level optimized for ease of use (timeline sequencer) to the second level that allows for more expressiveness (node graph). The design of our evaluation reflected this intention by first asking the participants to reproduce animations before they can freely create their own. P6's comment "the node graph becomes useful once I am intermediate level with the tool" confirms this intended transition.

9 LIMITATIONS AND FUTURE WORK

Although participants overall reacted positively to our system, they also provided constructive feedback and mentioned limitations for future improvements. P1 suggested that the animation curve of the Animate node could be integrated visually into the timeline sequencer in order to better edit the timing of animations. P5 wished

that more preset stylizations were provided, such as shape morphing. In the future, we could collect 3D motion graphics and extract commonly used motion effects from them to enrich our preset library. Although our tool works well for authoring animations, P2 wished that it could bake the animation data in an editable format for export so that they could render the animation in a more powerful renderer like Arnold [6] and Octane [44]. Moreover, replacing the node graph in our tool with a state machine [35] could support interactive actions for games and other applications instead of only pre-scripted animations.

Our layered authoring interface has two layers of abstraction. Adjusting the stylization parameters can be considered as the intermediate level between the timeline sequencer and the node graph since the user can in a lesser degree customize these effects. There are some interesting questions to consider about how to extend this layered representation. Would having more intermediate layers between the two levels ease the transitions between them? Can we add a layer beyond the second one (i.e. a deeper level that defines the behavior of the node graph), and how would that influence the balance between ease of use and expressiveness?

Our system has been designed with casual animation in mind to ease the barrier of entry for beginners. The clips in the timeline sequencer can be considered as a coarse version of keyframing (the start and end of a clip are two keyframes), serving as a starting point for learning more advanced keyframe controls in other tools. The implications of our design for learning are worth further study.

10 CONCLUSION

We have presented a system to help users easily create stylized 3D animations based on the traditional 2D animation principles [34, 55] that can adapt to common animation events such as contact and collision. Our layered authoring interface, consisting of a timeline sequencer and a node graph, balances between ease of use and expressiveness. A user evaluation and sample outcomes support the promise of our system, with potential future work in further enhancing the learning and authoring of 3D computer animations.

ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers for their valuable feedback as well as support from the following people (in alphabetical order of last name): Amanda McCoy Bast, Kenji Endo, Monika Hedman, Erin Kim, Timothy Langlois, Jeanette Mathews, Barbara Meier, Victoria Nece, Mike Ravella, Allison Yeh, and Agatha Yu.

REFERENCES

[1] Adobe. 2020. PhysicsWhiz | Adobe MAX Sneaks 2020. https://youtu.be/rutAGoBZtew
[2] Adobe. 2021. *Aero.* https://www.adobe.com/products/aero.html
[3] Adobe. 2021. *After Effects.* https://www.adobe.com/products/aftereffects.html
[4] Adobe. 2021. *XD.* https://www.adobe.com/products/xd.html
[5] Rahul Arora, Rubaiat Habib Kazi, Danny M. Kaufman, Wilmot Li, and Karan Singh. 2019. MagicalHands: Mid-Air Hand Gestures for Animating in VR. 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, 463–477. https://doi.org/10.1145/3332165.3347942
[6] Autodesk. 2021. *Arnold.* https://www.arnoldrenderer.com/
[7] Autodesk. 2021. *Maya.* https://www.autodesk.com/products/maya/overview
[8] Yunfei Bai, Danny M. Kaufman, C. Karen Liu, and Jovan Popović. 2016. Artist-Directed Dynamics for 2D Animation. *ACM Trans. Graph.* 35, 4, Article 145 (jul 2016), 10 pages. https://doi.org/10.1145/2897824.2925884
[9] Ronen Barzel, John F. Hughes, and Daniel N. Wood. 1996. Plausible Motion Simulation for Computer Graphics Animation. In *Proceedings of the Eurographics Workshop on Computer Animation and Simulation '96* (Poitiers, France). Springer-Verlag, Berlin, Heidelberg, 183–197.
[10] Andy Beane. 2012. *3D Animation Essentials.* John Wiley & Sons, Inc., Indianapolis, IN.
[11] Miklós Bergou, Saurabh Mathur, Max Wardetzky, and Eitan Grinspun. 2007. TRACKS: Toward Directable Thin Shells. *ACM Trans. Graph.* 26, 3 (jul 2007), 50–es. https://doi.org/10.1145/1276377.1276439
[12] Jonathan Blow. 2004. Game Development: Harder Than You Think: Ten or Twenty Years Ago It Was All Fun and Games. Now It's Blood, Sweat, and Code. *Queue* 1, 10 (feb 2004), 28–37. https://doi.org/10.1145/971564.971590
[13] Thor Brigsted. 2021. xNode. https://github.com/Siccity/xNode
[14] Xiang Cao. 2018. Wonder Painter: Turn Anything into Animation. In *ACM SIGGRAPH 2018 Real-Time Live!* (Vancouver, British Columbia, Canada) *(SIGGRAPH '18)*. Association for Computing Machinery, New York, NY, USA, Article 10, 1 pages. https://doi.org/10.1145/3229227.3229228
[15] Cascadeur Research. 2020. Physics In Animation. https://80.lv/articles/cascadeur-research-physics-in-animation
[16] Stelian Coros, Sebastian Martin, Bernhard Thomaszewski, Christian Schumacher, Robert Sumner, and Markus Gross. 2012. Deformable Objects Alive! *ACM Trans. Graph.* 31, 4, Article 69 (jul 2012), 9 pages. https://doi.org/10.1145/2185520.2185565
[17] Derivative. 2021. *TouchDesigner.* https://derivative.ca/
[18] Marek Dvorožňák, Daniel Sýkora, Cassidy Curtis, Brian Curless, Olga Sorkine-Hornung, and David Salesin. 2020. Monster Mash: A Single-View Approach to Casual 3D Modeling and Animation. *ACM Trans. Graph.* 39, 6, Article 214 (nov 2020), 12 pages. https://doi.org/10.1145/3414685.3417805
[19] Haegwang Eom, Byungkuk Choi, Kyungmin Cho, Sunjin Jung, Seokpyo Hong, and Junyong Noh. 2020. Synthesizing Character Animation with Smoothly Decomposed Motion Layers. In *Computer Graphics Forum*, Vol. 39. 595–606. https://doi.org/10.1111/cgf.13893
[20] Kenny Erleben, Jon Sporring, Knud Henriksen, and Henrik Dohlmann. 2005. *Physics-based animation.* Charles River Media, Hingham, MA, USA.
[21] Blender Foundation. 2021. *Blender.* https://www.blender.org/
[22] Martin Guay, Rémi Ronfard, Michael Gleicher, and Marie-Paule Cani. 2015. Space-Time Sketching of Character Animation. *ACM Trans. Graph.* 34, 4, Article 118 (jul 2015), 10 pages. https://doi.org/10.1145/2766893
[23] Thomas Huk, Mattias Steinke, and Christian Floto. 2003. Computer Animations as Learning Objects: What is an Efficient Instructional Design, and for whom?. In *Proceedings of the IADIS International Conference WWW/Internet 2003* (Algarve, Portugal) *(ICWI '03)*. IADIS, 1187–1190. http://www.iadisportal.org/digital-library/computer-animations-as-learning-objects-what-is-an-efficient-instructional-design-and-for-whom
[24] Khaled Hussein. 2020. Squash - Stretch Maker for Maya. https://khaledhussein.gumroad.com/l/AuyUu
[25] Jennifer Jacobs, Joel Brandt, Radomír Mech, and Mitchel Resnick. 2018. *Extending Manual Drawing Practices with Artist-Centric Programming Tools.* Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3173574.3174164
[26] Jaewoong Jeon, Hyunho Jang, Soon-Bum Lim, and Yoon-Chul Choy. 2010. A sketch interface to empower novices to create 3D animations. *Computer Animation and Virtual Worlds* 21, 3-4 (2010), 423–432. https://doi.org/10.1002/cav.353 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/cav.353
[27] Ming Jin, Dan Gopstein, Yotam Gingold, and Andrew Nealen. 2015. AniMesh: Interleaved Animation, Modeling, and Editing. *ACM Trans. Graph.* 34, 6, Article 207 (oct 2015), 8 pages. https://doi.org/10.1145/2816795.2818114
[28] Ben Jones, Jovan Popovic, James McCann, Wilmot Li, and Adam Bargteil. 2015. Dynamic Sprites: Artistic Authoring of Interactive Animations. *Comput. Animat. Virtual Worlds* 26, 2 (mar 2015), 97–108. https://doi.org/10.1002/cav.1608
[29] Benjamin James Jones. 2015. *Artist-guided Physics-based Animation.* Ph.D. Dissertation. Salt Lake City, UT, USA. Advisor(s) Bargteil, Adam Wade.
[30] 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* (Honolulu, Hawaii, USA) *(UIST '14)*. Association for Computing Machinery, New York, NY, USA, 395–405. https://doi.org/10.1145/2642918.2647375
[31] 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
[32] Rubaiat Habib Kazi, Tovi Grossman, Cory Mogk, Ryan Schmidt, and George Fitzmaurice. 2016. *ChronoFab: Fabricating Motion.* Association for Computing Machinery, New York, NY, USA, 908–918. https://doi.org/10.1145/2858036.2858138
[33] Rubaiat Habib Kazi, Tovi Grossman, Nobuyuki Umetani, and George Fitzmaurice. 2016. Motion Amplifiers: Sketching Dynamic Illustrations Using the Principles of 2D Animation. 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, 4599–4609. https://doi.org/10.1145/2858036.2858386
[34] John Lasseter. 1987. Principles of Traditional Animation Applied to 3D Computer Animation. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '87)*. Association for Computing Machinery, New York, NY, USA, 35–44. https://doi.org/10.1145/37401.37407
[35] Germán Leiva, Jens Emil Grønbæk, Clemens Nylandsted Klokmose, Cuong Nguyen, Rubaiat Habib Kazi, and Paul Asente. 2021. *Rapido: Prototyping Interactive AR Experiences through Programming by Demonstration*. Association for Computing Machinery, New York, NY, USA, 626–637. https://doi.org/10.1145/3472749.3474774
[36] Long Bunny Labs. 2017. Squash & Stretch Kit - Effect Component for Unity. https://youtu.be/RjFjx_sC_XM
[37] Jacques Lucke. 2021. Animation Nodes for Blender. https://animation-nodes.com/
[38] Luxion. 2021. *KeyShot*. https://www.keyshot.com/
[39] Maxon. 2021. *Cinema 4D*. https://www.maxon.net/en/cinema-4d
[40] Microsoft. 2021. *PowerPoint*. https://www.microsoft.com/en-us/microsoft-365/powerpoint
[41] Matthew Moore and Jane Wilhelms. 1988. Collision Detection and Response for Computer Animation. (1988), 289–298. https://doi.org/10.1145/54852.378528
[42] Brad A. Myers. 1990. Taxonomies of Visual Programming and Program Visualization. *J. Vis. Lang. Comput.* 1, 1 (mar 1990), 97–123. https://doi.org/10.1016/S1045-926X(05)80036-9
[43] New Frame Plus. 2020. The 12 Principles of Animation in Games. https://www.youtube.com/playlist?list=PLugegG07di3886WYN6u7v9BeBd0VFG3_J
[44] OTOY. 2021. *OctaneRender*. https://home.otoy.com/render/octane-render/
[45] Mengqi Peng, Jun Xing, and Li-Yi Wei. 2018. Autocomplete 3D Sculpting. *ACM Trans. Graph.* 37, 4, Article 132 (jul 2018), 15 pages. https://doi.org/10.1145/3197517.3201297
[46] Nazmus Saquib, Rubaiat Habib Kazi, Li-Yi Wei, and Wilmot Li. 2019. *Interactive Body-Driven Graphics for Augmented Video Performance*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3290605.3300852
[47] Johannes Schmid, Robert W. Sumner, Huw Bowles, and Markus Gross. 2010. Programmable Motion Effects. , Article 57 (2010), 9 pages. https://doi.org/10.1145/1833349.1778794
[48] Francis Schmidt, Jim Jagger, Jim McCambell, and Craig Slagel. 2004. 3D Animation: Difficult or Impossible to Teach and Learn?. In *ACM SIGGRAPH 2004 Panels* (Los Angeles, California) *(SIGGRAPH '04)*. Association for Computing Machinery, New York, NY, USA, 3. https://doi.org/10.1145/1186554.1186557
[49] Thomas W. Sederberg and Scott R. Parry. 1986. Free-Form Deformation of Solid Geometric Models. In *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '86)*. Association for Computing Machinery, New York, NY, USA, 151–160. https://doi.org/10.1145/15922.15903
[50] 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
[51] Side Effects Software. 2021. *Houdini*. https://www.sidefx.com/products/houdini
[52] Tiago Boldt Sousa. 2012. Dataflow programming concept, languages and applications. In *Doctoral Symposium on Informatics Engineering*, Vol. 130.
[53] Unity Technologies. 2021. *Unity*. https://unity.com/
[54] Unity Technologies. 2021. *Unity Timeline*. https://unity.com/features/timeline
[55] Frank Thomas and Ollie Johnston. 1995. *The Illusion of Life: Disney Animation*. Disney Editions, New York, NY, USA.
[56] John Thompson, Zhicheng Liu, Wilmot Li, and John Stasko. 2020. Understanding the Design Space and Authoring Paradigms for Animated Data Graphics. *Computer Graphics Forum* 39, 3 (2020), 207–218. https://doi.org/10.1111/cgf.13974 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.13974
[57] Anh Truong, Peggy Chi, David Salesin, Irfan Essa, and Maneesh Agrawala. 2021. Automatic Generation of Two-Level Hierarchical Tutorials from Instructional Makeup Videos. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) *(CHI '21)*. Association for Computing Machinery, New York, NY, USA, Article 108, 16 pages. https://doi.org/10.1145/3411764.3445721
[58] Jue Wang, Steven M. Drucker, Maneesh Agrawala, and Michael F. Cohen. 2006. The Cartoon Animation Filter. *ACM Trans. Graph.* 25, 3 (jul 2006), 1169–1173. https://doi.org/10.1145/1141911.1142010
[59] Chris West. 2021. MegaFiers 2. http://www.west-racing.com/mf/?page_id=2
[60] 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
[61] Lei Zhang and Steve Oney. 2020. *FlowMatic: An Immersive Authoring Tool for Creating Interactive Scenes in Virtual Reality*. Association for Computing Machinery, New York, NY, USA, 342–353. https://doi.org/10.1145/3379337.3415824

A DIFFERENCE FROM MOTION AMPLIFIERS [33]

Although easy to use, Motion Amplifiers has limited expressiveness due to its lack of a global timeline and customization capability. The user can only work with the preset effects and cannot sequence different animations and effects. This has confined the output of Motion Amplifiers to looping animations without coordination among objects. The timeline sequencer with movable clips in our system allows for complex compositing and coordination, and the node graph enables the creation of custom effects. In terms of modularity, each motion effect in Motion Amplifier is independent of each other and can be toggled on or off. Our tool offers more fine-grained modularity than the binary on-or-off; the user can use the timeline sequencer to decide when and how long they want certain effects to take place, gaining more control and freedom over the outcome. Moreover, Motion Amplifier does not handle events such as contact and collision, which is essential for 3D animations [41]. In our tool, collisions are detected at every frame and indicated to the user via event markers to facilitate the positioning of clips and fine-tuning of stylizations.

Overall, our system more effectively balance between ease of use and expressiveness through the layered authoring interface paradigm and, as a result, can produce a wider variety of animations as shown in Figures 8 and 9.