# Triggering Artwork Swaps for Live Animation

**Nora S Willett***, **Wilmot Li**†, **Jovan Popović**†, **Adam Finkelstein***

*Princeton University †Adobe Research


**Figure 1: Preparing a character for live animation. First, the user creates a layered illustrated character with triggerable artwork. Next, in our interface, the user selects from default layouts (row layout shown) or rearranges triggers to create their own. Then, the user practices their performance. Finally, during the live performance, our system assists the user in selecting the next pose using a predictive triggering model trained on the practices.**

## ABSTRACT

Live animation of 2D characters is a new form of storytelling that has started to appear on streaming platforms and broadcast TV. Unlike traditional animation, human performers control characters in real time so that they can respond and improvise to live events. Current live animation systems provide a range of animation controls, such as camera input to drive head movements, audio for lip sync, and keyboard shortcuts to trigger discrete pose changes via artwork swaps. However, managing all of these controls during a live performance is challenging. In this work, we present a new interactive system that specifically addresses the problem of triggering artwork swaps in live settings. Our key contributions are the design of a multi-touch triggering interface that overlays visual triggers around a live preview of the character, and a predictive triggering model that leverages practice performances to suggest pose transitions during live performances. We evaluate our system with quantitative experiments, a user study with novice participants, and interviews with professional animators.

### Author Keywords

2D animation; live performance

### ACM Classification Keywords

H.5.2 Information interfaces and presentation: User interfaces - Graphical interface.

## INTRODUCTION

To date, most animation is produced *offline*: an artist painstakingly crafts the animation, which is later viewed in the form of movies, television shows, promotions, advertisements, etc. Recent advances in performance-based animation combined with the increase of live streaming platforms have given rise to a new form of *live* animation, where performers control animated characters for live audiences. Last year, Homer answered live phone calls during an episode of The Simpsons [8], Stephen Colbert interviewed cartoon guests on The Late Show [12], and animated characters hosted live chat sessions on Facebook and YouTube [32, 33]. In live animation, characters interact with and respond to live audiences in ways that have not been possible with offline animation.

A key challenge in live animation is the mapping of human action onto digital characters in real time. Motion capture with face and pose tracking enables literal translation of performed movement onto parts of the character but disregards other problems, including dialogue, anticipation, timing, exaggeration, and secondary effects. Animated dialogue, for example, relies on discrete transitions to approximate mouth shapes instead of detailed lip tracking. Precise timing, expressive gestures, and exaggerated poses are also better accomplished with discrete transitions instead of smooth changes. This preference is true across many styles of animation, including stop-motion, computer, 3D, or 2D animation.

This work examines 2D animation with characters represented by a set of artwork layers that typically depict different body parts or accessories (e.g., hats, clothes). Many cartoons are created in this style, including the live animation examples shown in Figure 2. Within this context, discrete transitions are realized by swapping artwork layers to produce large changes in pose and appearance. For example, gestures such as shrugging, pointing and fist shaking, are often portrayed as alternative artwork layers for arms and hands. Similarly, exaggerations like a character's eyes turning into hearts and spirals, or thought bubbles appearing overhead are also encoded in this way.

While discrete artwork swaps increase the appeal and expressiveness of 2D animated characters, only some types of swaps,


*UIST 2017, October 22–25, 2017, Qubec City, QC, Canada.*

ACM ISBN 978-1-4503-4981-9/17/10 ...$15.00.
http://dx.doi.org/10.1145/3126594.3126596

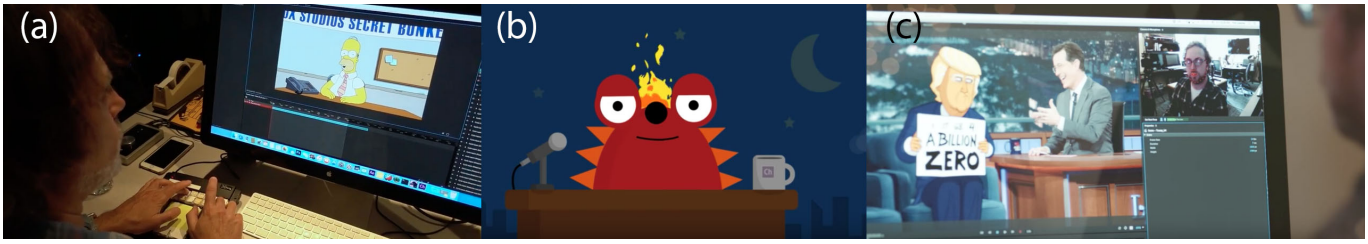


**Figure 2: Live animation examples: (a) Homer – The Simpsons [8]. (b) Red Monster – YouTube Live [32]. (c) Cartoon Trump – The Late Show [12].**

such as triggering mouth shapes based on an input audio stream, are easily automated. Most other types present a challenge for human performers, especially given the goal of triggering correct swaps at correct times throughout a live performance. In interviews with live animation producers, we learned that the most common setup involved the use of keyboard shortcuts to trigger swaps. In this setting, the performer must not only remember all the shortcuts but also decide which swaps to execute in real time. Moreover, most live performances require the character to talk as well, which adds even more cognitive load to the task. Due to these challenges, some of the production teams we talked to (e.g., The Simpsons) use two or more performers for a single character.

In this work, we design and evaluate a multi-touch interface for triggering artwork swaps in a live animation setting. Our approach leverages two key insights. First, to help users execute swaps more efficiently, our interface arranges visual triggers that show thumbnail images of the corresponding artwork around a live preview of the character. This trigger ***Layout*** design enables users to quickly recognize and tap triggers without looking away from the character. Second, since animators typically practice before live performances, we encode common patterns from practice sessions in a predictive model that we then use to highlight suggested triggers during performances. Our highlighted ***Predictions*** assist performers in executing common sequences without preventing them from improvising.

We used our system to create performances with five different characters, which are included in the accompanying materials for our paper. In addition, we evaluated our approach in several other ways. We analyzed the performance of our predictive model with quantitative experiments using existing live animation sequences from The Late Show as ground truth data. We also conducted a comparative user study with mostly novice participants where we found a strong preference for our multi-touch interface over keyboard triggers and that suggestions facilitate the triggering task. Finally, we solicited professional feedback from the production team at The Late Show based on an informal demo session with our system.

## RELATED WORK

We discuss related previous work on performed animation, interfaces for text input, and predictive user interfaces.

### Performance-Based Animation

Previous performance-based animation systems explore a variety of techniques for capturing human performances. Many existing methods leverage continuous input, including motion capture of the body [36] and face [42], puppetry with physical props and other sensors [2, 7, 16, 25, 28], direct manipulation via touch [15, 23], and sketching [5, 13, 17]. In contrast, we focus on the task of triggering discrete artwork changes. Some existing animation tools support automated triggering driven by specific types of input, such as swapping mouth shapes based on speech (i.e., lip synchronization) [1, 31, 38] and triggering pre-defined facial expressions based on the performer's appearance [35, 43]. However, general-purpose triggering is typically performed manually via keyboard shortcuts. We propose a new multi-touch interface that is more effective and configurable than keyboard triggering.

### Multi-touch Triggering

Previous work has explored multi-touch triggering interfaces in other domains. Given that text input is a form of discrete triggering, our problem is related to the design of soft keyboards for touch devices [10, 44]. As with keyboard layout, we aim to generate a trigger arrangement that helps users hit the desired sequence of discrete targets. Researchers have also explored the design of multi-touch DJ and VJ interfaces [14, 19], which share our focus on live performances. However, the problem of arranging animation triggers presents unique challenges. First, the set of animation triggers varies from one character to the next, which means that our system must adapt appropriately. Moreover, in our pilot study, we found that users prefer triggers to be near the position of the corresponding triggered artwork, which puts additional spatial constraints on the layout. In this respect, our approach is related to label layout algorithms (e.g. [3, 4, 39]) that encourage labels to appear close to the corresponding anchor regions.

### Predictive User Interfaces

Our interface leverages an animator's practice sessions to learn a probabilistic model that predicts the next trigger during a performance. In this respect, our system is an example of a predictive or adaptive user interface. Previous work has explored this idea across several domains, including app or icon selection on mobile interfaces [30, 41], dialog box interaction [11], menu navigation [9], text input [10], and command selection [6, 22]. We investigate the use of predictive models in the specific task of live animation triggering. In particular, we apply Markov models in a similar manner as previous command prediction systems [6] to encode and predict common patterns of triggers.

## DESIGN STUDY

To better understand the needs of live animators, we interviewed several authors: the production team for the live Simpsons episode [8] in which Homer spent three minutes responding to phone-in questions from viewers; the animators at The Late Show who have worked on roughly 50 live animation performances, ranging in length from 30s to six minutes, with seven different characters [12]; the author of a 30 minute question-and-answer session on YouTube live featuring a cartoon monster [32]; and the host of a Twitch show that includes a live cartoon avatar who provides real-time commentary during streaming gameplay sessions [34].

All animators used triggered artwork swaps extensively in their performances, and they noted several common challenges with the current triggering workflow. First, it is difficult to remember the mapping from keys or buttons to the corresponding artwork swap. To this end, the Simpsons and Late Show teams added physical annotations (images and drawings) to the keys/buttons of their triggering devices (Figure 3). Even with these visual cues, animators find it challenging to survey the potential options and then execute the selected triggers at exactly the right time, especially when improvising. In fact, the Simpsons and Late Show dedicated one performer to operating the triggers while others handled the speaking. This option is obviously not available to individual animators who produce content by themselves. Finally, since these are typically live performances (in some cases, broadcast to millions of viewers!) with no opportunity for re-takes or post production, minimizing mistakes is paramount.

Based on these discussions, we identified the following design goals for a live performance triggering interface:

***Intuitive.*** The mapping between triggers and the corresponding artwork should be explicit so that performers do not have to memorize a large number of triggers.

***Accurate.*** The triggering interaction itself should be predictable and accurate to minimize mistakes.

***Fast.*** The interaction should also be fast to help animators coordinate swaps with the rest of their performance.

***Accessible.*** To cover the broadest range of use cases, the system should not require highly specialized input devices or multiple performers for a single character.

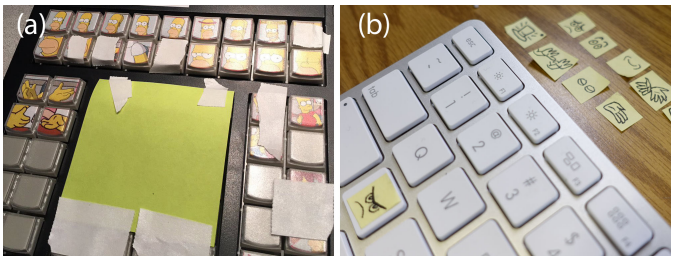


**Figure 3: Setups used for live performances: (a) The Simpsons [8]. (b) The Late Show [12].**

## OUR APPROACH

Given these design goals, we create an interface for triggering artworks swaps for live animation. An important high-level design decision for our system is what type of input device to support. We considered several options:

***Keyboards*** are very accessible, but the mapping between keys and triggers is not intuitive, as noted in the previous section. While physical annotations (as in Figure 3) can help, reconfiguring them for characters with different sets of triggers (or for normal keyboard usage) is inconvenient.

***Game controllers*** are designed to be responsive, but as with keyboards, the mapping between input interactions and triggers is not intuitive.

***Voice commands*** are arguably a more intuitive triggering interaction than pressing abstract keys or buttons. However, animators typically need to include speech as part of their performances.

***Real-time tracking*** of hand and body poses is another potential input modality. Video-based techniques are the most accessible, but even state-of-the-art algorithms like Wei *et al.*[40] are not accurate enough (88%) for our application. Techniques that rely on depth data (e.g., [21, 24, 26, 29, 37]) are more accurate, but depth cameras are still far less common than regular cameras. Marker-based motion capture is the most accurate but also the most obtrusive. Moreover, even perfect tracking requires mapping a continuous input signal (e.g., joint angles, positions) to a discrete set of triggers, which raises its own set of challenges.

Ultimately, we decided to design our triggering interface using ***multi-touch displays***, which are widely available (primarily in the form of tablets) and provide a reconfigurable, interactive display surface where we can render triggers. To make the mapping between triggers and artwork intuitive, we show triggers with thumbnail images of the corresponding artwork as a visual cue. To make interactions fast and accurate, we arrange the triggers around a live preview of the character (Figure 1) so that performers can view and tap triggers quickly and accurately without looking away from the character. Our prototype uses Adobe Character Animator [1] as the real-time rendering engine for the character, which allows us to support face tracking and audio-driven lip sync in addition to the core triggering functionality of our system.

The key features of our interface design are informed by two important observations from our pilot study with an early version of the system. The first is that the layout of the triggers have a significant impact on how quickly users are able to find triggers and how much they improve with practice. Moreover, users have a very hard time deciding or remembering which triggers to hit during a performance, even after practicing several times. Based on these observations, we developed an automatic trigger ***Layout*** algorithm that arranges triggers according to the spatial distribution of the corresponding artwork and a ***Prediction*** feature that helps animators select the appropriate triggers at performance time. The following sections describe the types of available triggers and these two components in detail.

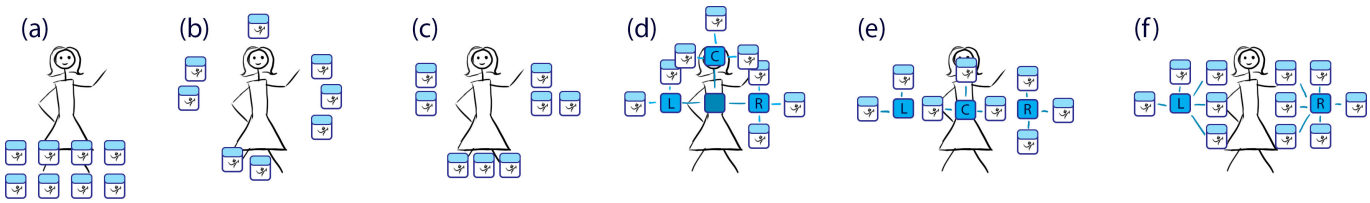


**Figure 4: Sketches of the different layout designs we prototyped. (a) Grid layout. (b) Radial layout. (c) Clustered layout. (d) FMM. (e) PMM. (f) SMM.**

## TRIGGERS

When a trigger is activated, it can cause a simple artwork swap or a cycle swap. Simple swaps involve one piece of artwork changed for another. Cycle swaps invoke short looping animations or smooth transitions which hold on a final extreme pose. A swap can also have sub-poses which are triggered only once the main pose is already active. Sub-poses modify a main pose by either deforming it slightly (moving up, down left or right) or switching to similar artwork (holding up one, two or three fingers)

Our interface implements three types of triggers. The first is the single selection trigger shown in blue (Figure 5a). The user presses this trigger with one finger causing the character to perform a swap. The next type of trigger is numerical selection (red) which triggers a separate swap depending on the number of fingers pressed down in its vicinity (Figure 5b). This trigger is typically used for displaying different numbers of fingers on a hand or different pointing styles. The third trigger, radial selection (orange), allows the user to trigger sub-poses by swiping their finger in the space around the trigger (Figure 5c). The area around the trigger is divided into radial slices based on the number of sub-poses. The area of the visual trigger does not trigger a sub-pose. Radial selection is useful to simulate continuous movement with each slice triggering predeformed artwork. It can also replicate a cycle with the user controlling the speed by how quickly they swipe around the trigger. When any trigger is activated, the border becomes thicker and the color darker (Figure 5d). Please see our submission video for examples of the trigger types in action.

For each type of trigger, the user can also control how long a trigger stays active after released. This "rest delay" allows directly transitioning between poses without automatically switching to the rest pose in between. To indicate the amount of time left before the trigger releases, a counting down circle appears (Figure 5d).

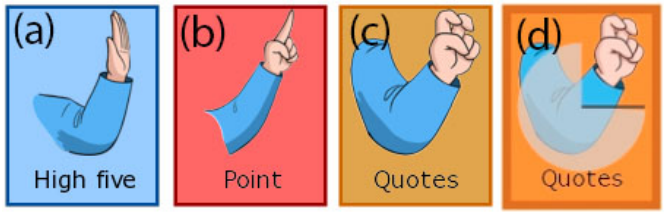


**Figure 5: The three types of triggers: (a) single selection, (b) numerical selection, and (c,d) radial selection. Our visualization for the decreasing rest delay of a trigger is also shown in (d).**

## LAYOUT

We examine various designs for potential trigger layouts and present an automatic layout algorithm for creating our final designs.

### Design

To explore the space of potential trigger layouts, we prototyped several different designs, as shown in Figure 4. For these initial prototypes, we hard-coded the positions of the triggers for a given character.

***Display all.*** Some designs display all the triggers in a fixed layout. The simplest design arranges triggers in a uniform grid below the character (Figure 4a). The radial layout positions triggers around the center of the character, which makes better use of screen space than the grid (Figure 4b). We also implemented a clustered layout that groups together related triggers and positions them in different locations around the character (Figure 4c).

***Hierarchical.*** One drawback of displaying all the triggers is that the interface may become cluttered as the number of triggers increases. Thus, we also prototyped marking menu layouts [18, 20] that hide triggers within the menu hierarchy. In the full marking menu layout (FMM), we first organize triggers into three groups (left, right, center) based on the position of the artwork relative to the vertical center line of the character, and then we merge these groups into a single top-level group, which is the only trigger that is visible by default (Figure 4d). The user explores the hierarchy by first touching the top-level group, swiping to one of the mid-level groups, and then swiping to one of the individual triggers. To reduce the depth of the hierarchy, we also created a positioned marking menu (PMM) where the left, right and center groups are visible at the top level. Finally, we created a split marking menu (SMM) that only has top-level left and right groups. To reveal the center triggers, the user selects both left and right groups at the same time (with two separate fingers), and then swipes either finger to hit the desired target.

***Findings.*** We compared the radial, clustered, PMM, and SMM layouts in a pilot study with nine participants who recorded performances with all four designs. We decided not to include the grid and FMM layouts, since our own informal tests suggested that these two designs were inferior to the others. Overall, users were more effective with the display all layouts. Swiping through even two levels of hierarchy with the marking menus introduced significantly more latency than the radial and clustered designs, even after a fair amount of practice. Regarding the actual layouts, users voiced strong objections to any designs that positioned triggers over the character (even

when the occluding triggers only appeared occasionally, during a swiping interaction). Participants favored layouts where the trigger positions matched the positions of the artwork (e.g., "hands up" triggers near the top and "hands down" triggers near the bottom of the character). They also liked layouts where triggers for corresponding left-right poses (e.g., "left fist" and "right fist") were positioned symmetrically around the character. In the end, users indicated a slight preference for the clustered layout over the radial design. They felt that the radial layout was too cluttered with all the triggers so close to each other and the character.

### Proposed Algorithm

These findings led us to design an automatic method for generating clustered layouts. Our algorithm recommends triggers be placed in layout slots that are close to the trigger artwork. Given a character with a set of triggers $T$ along with the corresponding set of artwork, we formulate trigger layout as a variant of the assignment problem. Based on the total number of triggers $|T|$, we first define a set of layout slots $S$. By default, if $|T| \leq 43$, we create a 4x4 grid of slots on either side of the character and a 1x11 row of slots below the character. If $|T| > 43$, we increase the row and column count for each grid and scale down the size of the slots to fit the screen until there are enough slots for all the triggers. We did not encounter a character with more than 25 triggers. Based on feedback from the pilot study, we define both a **row** layout where the left and right grids are completely regular, and a **fan** layout where each row of slots is bent into a shallow fan that better matches the natural resting position of fingers on the display (Figure 6).

For either layout type, the goal is to assign each trigger $t \in T$ to a unique layout slot $s \in S$ such that the overall assignment $A$ minimizes a cost function

$$C(A) = \sum_{t,s \in A} c(t, s)$$

where the per-assignment cost $c(t, s)$ is defined as the distance between the centroid of the artwork for trigger $t$ and the centroid of the layout slot $s$. We compute these centroids in a consistent coordinate frame where the origin is at the center of the screen, and the artwork is translated such that the centroid of the character's designated rest pose (i.e., the pose that the character starts in and returns to when no other triggers are active) is at the origin. This cost function penalizes assignments that put triggers far away from their corresponding artwork. To find the optimal assignment, we use the Munkres


**Figure 6: Automatically generated row (a) and fan (b) layouts. The dotted boxes are positions for triggers that have not been filled.**

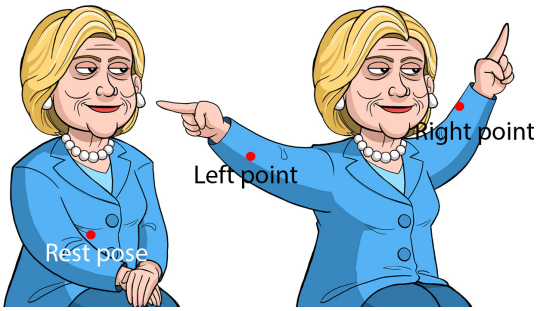


**Figure 7: Left-right triggers for pointing which are not symmetric with respect to the rest pose. The red circles are the centroids of the artwork.**

algorithm [27]. Afterwards, we render each trigger in the assigned layout slot and use a scaled down version of the trigger artwork as the thumbnail image.

While the algorithm as described encourages triggers to be assigned to layout slots that are close to the trigger artwork, we found that two small modifications to the cost function helped produce better results.

***Left-right triggers.*** With the above cost function, we observed that corresponding left-right triggers did not always end up in symmetric slots. This mishap is because the assignment of any given trigger depends globally on all other triggers. Moreover, depending on how the character is drawn, the artwork for left-right triggers may not actually be symmetric with respect to the character's rest pose (Figure 7). To address this problem, for each pair of left-right triggers $(t_L, t_R)$, we compute the cost of assigning $t_L$ to each lefthand slot and $t_R$ to each righthand slot, average the costs for symmetric slots $s_L$ and $s_R$, and then set the costs for $s_R$ to the average. We compute an optimal assignment for all $t_R$ and assign slots $s_R$ from that computation. We assign slots $s_L$, not based on the computation, but by forcing $s_L$ to be symmetric to $s_R$. Afterwards, we process the remaining triggers. We identify left-right triggers by looking for matching left and right tags (e.g., "left point" and "right point").

***Both-side triggers.*** Subjects in our pilot study preferred that triggers involving both arms, hands or legs (e.g., crossed arms, clasped hands) were arranged in a row below the character. To encourage such layouts, we artificially set the y-coordinate of the artwork centroids for such both-side triggers to match the y-coordinate of the bottom row layout slots, which reduces the penalty for assigning the triggers to that row. Such triggers were found by searching for tags that include the word "both."

## PREDICTION

Another finding from our pilot study was that participants consistently had trouble deciding or remembering what triggers to use at performance time. In particular, all users found it hard to focus on triggering while they were speaking. We address this problem with prediction, by training a probabilistic model that learns common patterns from practice sessions. At performance time, we query our model to predict the most likely subsequent triggers and then highlight these predictions to the user as hints.

### Model

We represent each practice session with a sequence of trigger states $(q_1, q_2, \cdots, q_N)$. Each state $q$ is a $|T|$-dimensional bit vector representing the status of every trigger, which is either on (i.e., the user is touching the trigger and the corresponding artwork is visible) or off. For each practice session, we record the state after every touch down and touch up event. To ensure that "simultaneous" trigger events (e.g., triggering "left hand up" and "right hand up" at the same time) are treated as a single state change rather than two changes in rapid succession, we cluster states that occur within 80ms of each other and only keep the last one. Empirically, we found that 80ms was long enough to account for simultaneous triggers while still capturing quick transitions.

Markov models are a natural choice for encoding transition probabilities between trigger states. However, modeling transitions between individual states $q$ does not provide enough context to give useful predictions. In most cases, we need to analyze a longer history of trigger states to accurately predict what the next state might be. In other words, rather than working with individual states, we must consider sequences of states $(q_{i-n}, q_{i-n+1}, \cdots, q_i)$ (or $n$-grams where $n$ is the length of the sequence) when building the model. On the other hand, large $n$-grams dramatically increase the state space (i.e., all possible $n$-grams) for the model, making it very likely that we will encounter $n$-grams at performance time that have never been observed in the practice data.

Our solution is to construct an ensemble of Markov models, each of which uses different sized $n$-grams in the range $[n_{\text{min}}, n_{\text{max}}]$. At performance time, given the current trigger state $q$, we compute a weighted combination of the transition probabilities from every model to determine the most likely next states. This approach allows us to make use of as much context as possible while still producing valid predictions based on the more immediate trigger history. We experimented with four different weighting schemes that set the weight for each Markov model as follows: 1) the frequency of the current $n$-gram normalized by the total number of $n$-grams in the model's training data; 2) a uniform value; 3) $(n_{\text{max}} - n)/(n_{\text{max}} - 1)$; 4) $(n-1)/(n_{\text{max}} - 1)$. The first weighting scheme favors Markov models that have more training data that matches the current state; the second model weights all models equally; and the third and fourth models give decreasing and increasing weight to longer $n$-grams. We set $n_{\text{min}}$ to be 2 because 1-grams are not very informative in our setting; most of the time, a triggered event is followed by a return to the rest pose, so 1-grams just model the overall occurrence frequency of individual trigger states. We determine $n_{\text{max}}$ and the best weighting scheme empirically, as described in the Evaluation section.

### Highlighting Predictions

Given the set of next state probabilities from the predictive model, our system highlights the most likely triggers and greys out the rest (Figure 8). Users can choose whether the system highlights the top one or top three suggestions. When highlighting a trigger $t$, we change its size and saturation as a function of its probability $p(t)$ of being next. We scale $t$ by $(1 + p(t)/2)$, so that the maximum size is 50% larger than the original trigger size. Enlarging triggers any more causes overlaps. We compute the saturation of $t$ by linearly interpolating between its original color and grey, using the parameter $\alpha = p_{\text{max}} - p(t)$ where $p_{\text{max}}$ is the largest of all next trigger probabilities. Showing a single suggestion minimizes the amount of visual clutter but may omit useful suggestions when there is more than a single likely prediction. Showing three suggestions gives the user more options at the cost of more clutter. Users can also choose to turn off the suggestions entirely, in which case none of the triggers are greyed out. Note that users can tap on any trigger, even the grey ones.


**Figure 8: Our assisted performance design. In (a), the triggers with the highest probability of being selected next are highlighted. In (b), the user is signaled to lift all their fingers from the screen in order to return to the rest pose.**

## IMPLEMENTATION

We ran our system on a MacBook Pro, 2.5 GHz Intel Core i7, 16GB DDR3 using a Wacom Cintiq 13HD touch as our multi-touch device. To animate the character and render the live preview in our interface, we use Adobe Character Animator [1]. In addition to keyboard triggering of artwork swaps, Character Animator allows users to animate a character's facial features via real-time face tracking, and it automatically animates the character's mouth by analyzing the performer's speech. Our system converts a visual trigger press to a keyboard trigger event and passes it to Character Animator to activate the proper artwork swap. We implemented our system in C++ using OpenFrameworks[1], an open source C++ tool kit for creative coding. It provides a simple and intuitive framework for experimentation with graphical user interfaces. The OpenFrameworks environment supports an expanding list of addons which offer additional functionality. We use the following addons: ofxJson[2], ofxTransparentWindow[3], ofxWacomFeelMultiTouch[4] and ofxDatGui[5].

## EVALUATION

To validate our approach, we ran quantitative experiments on our predictive triggering model and conducted a user study comparing different modes of our interface to a baseline keyboard triggering setup. We also obtained informal feedback on our system from the production team at The Late Show.

[1] http://openframeworks.cc/
[2] https://github.com/jeffcrouse/ofxJSON
[3] https://github.com/jeffcrouse/ofxTransparentWindow
[4] https://github.com/andreasmuller/ofxWacomFeelMultiTouch
[5] https://github.com/braitsch/ofxDatGui

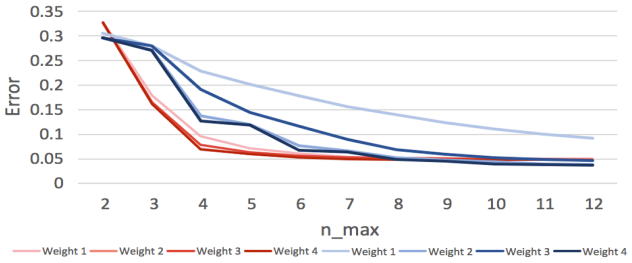


**Figure 9: Searching for the best combination of weight type and $n_{\max}$. Weight type 4 and $n_{\max} = 8$ offers the best behavior in both the Trump (red) and Hillary (blue) tests.**

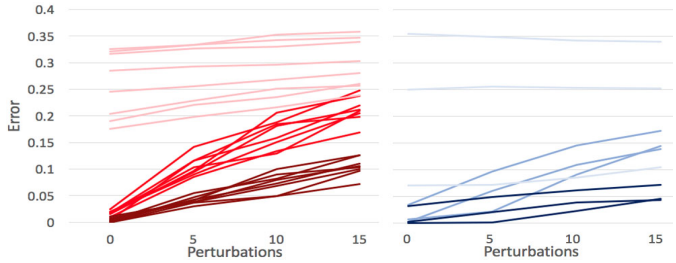


**Figure 10: In both Trump (red) and Hillary (blue) the darkest color uses $n_{\max} = 8$ and weight type 4. The medium and light colors use standard Markov models with 8-grams and 2-grams respectively.**

## Model Validation

We ran several experiments on a ground truth dataset of live animation performances in order to: (1) evaluate our predictive triggering model, (2) determine the best value for $n_{\max}$ (maximum $n$-gram length), and (3) choose an appropriate weighting scheme for our ensemble of Markov models. To obtain ground truth, we manually recorded the sequence of triggered poses for all the appearances of Cartoon Trump and Cartoon Hillary on The Late Show. There were 11 performances in total (eight for Trump and three for Hillary), ranging from 30 seconds to six minutes each of which contained between nine and 108 trigger events.

We measure the error $E$ for a given test sequence $Q = (q_1, q_2, \cdots, q_N)$ as follows:

$$E(Q) = \sum_{i=1}^{N} e(q_i), \quad \text{where} \quad e(q_i) = \begin{cases} 0, & \text{if } q_i \text{ is 1st choice} \\ 1, & \text{if } q_i \text{ is 2nd choice} \\ 2, & \text{if } q_i \text{ is 3rd choice} \\ 3, & \text{otherwise} \end{cases}$$

In our first experiment, we generate simulated practice sessions by randomly perturbing the original sequence for each performance. Specifically, we generate four practice runs where 0%, 5%, 10% and 15% of the original sequence is perturbed. To create a perturbed sequence at target perturbation level $L$, we insert a new state at each location in the original sequence with probability $L$. The new state is chosen randomly from the set of unique states for the character. Using this procedure, we generate ten sets of practice runs for each performance, which we use as training data. For testing, we generate 100 sequences at each of the four perturbation levels. In total, that leaves us with 10 training sets and 400 test sequences for each ground truth performance.

To determine the best weighting scheme and $n_{\max}$ for our model, we consider all combinations of our four weighting schemes with $n_{\max}$ values ranging from two to twelve. For every ground truth performance, we take each weighting scheme and $n_{\max}$ combination, train ten predictive models using each of our ten training sets, and run all 400 test sequences distributed evenly across the ten models. From these runs, we compute a cumulative error $E$. Figure 9 shows these errors for Trump and Hillary, averaged over all performances and normalized by the total sequence length. This data shows that weighting scheme four (where the weight increases for longer $n$-grams) yields the lowest error. Moreover, the error tends to plateau when $n_{\max}$ increases past eight or nine. As a result, we use weighting scheme four and $n_{\max} = 8$ for all of our results and the user study described below.

The second experiment measures the impact of combining an ensemble of Markov models in our predictive model. As baselines, we implemented two standard Markov models, one that uses 2-grams and one that uses 8-grams. With these two models, we compute the normalized error for the same set of training-test combinations described above and compare the results against our predictive model using the optimal weighting scheme and $n_{\max}$. Figure 10 plots the results for each performance at every test sequence perturbation level. Our ensemble model yields significantly lower errors than either of the standard Markov models, with the 2-gram model performing the worst by a considerable margin. Even at 0% perturbation, the 2-gram model produces higher errors than our predictive model at 15% perturbation for all performances.

As a final experiment, we evaluated the performance of our predictive model when we use different ground truth performances as training. In particular, we ran leave one out validation on the performances for each character, which simulates a scenario where animators use previous performances (rather than specific practice sessions) to train the model. Not surprisingly, given the discrepancies between the training and test sequences, the normalized error for each performance was significantly higher than in the previous experiments. In spite of this, we found that in six out of 11 performances, the error was less than two, which means that the next state was generally one of the top three suggestions.

## User Study

In addition to the quantitative evaluation of our predictive triggering model, we wanted to gather feedback on our system as a whole. To this end, we conducted a user study with 16 participants (14 novices and two experienced animators) comparing four triggering interfaces:

***Keyboard*** is the baseline condition where users trigger with keyboard shortcuts.

***NoSugg*** is our interface with no highlighted suggestions (i.e., no triggers are greyed out during performance).

***1Sugg*** is our interface with one highlighted suggestion.

***3Sugg*** is our interface with three highlighted suggestions.

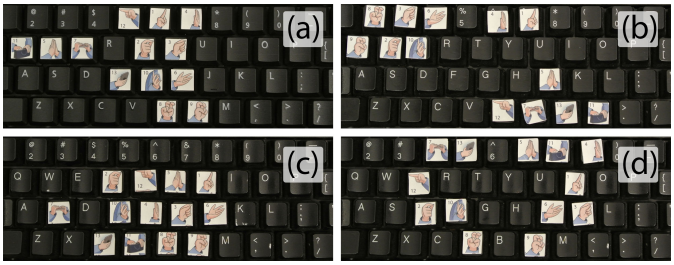


**Figure 11: (a) The default keybaord layout used in our final user study. (b,c,d) Some of the customized keyboard layouts created by users.**

We used a within-subject design where each participant used all four interfaces to animate Cartoon Hillary answering interview questions. Given a script of both the questions and responses, participants were asked to create live animations in the following four tasks:

1. Answer the questions in the script in order.
2. Answer the questions in the script in a different order.
3. Answer the questions in the script in order with one new question in the middle.
4. Answer similar questions as in the script but with slightly different wording for each.

The tasks simulate a typical live interview scenario where the interviewee is often provided with potential questions ahead of time. However, interviewers sometimes deviate from the script to varying degrees. Participants were free to decide which triggers to use during their performances. We fixed the order of the tasks from easiest (*Task 1*) to hardest (*Task 4*) and counterbalanced the order of the interface conditions using a 4x4 Latin square. With 16 participants, we had four users for each task-interface combination. Please see our supplemental materials for the scripts and some videos of participants performing the tasks.

***Setup.*** We ran our interface on a Wacom Cintiq 13-inch multitouch display. For the *Keyboard* condition, we placed stickers with the visual trigger icons on the keys (Figure 11a).

***Practice.*** Each study session consisted of two parts. During the practice period, participants rehearsed their scripted responses four times using the *Keyboard*. After the first practice run, we allowed users to customize the sticker placement. Seven decided to do so (Figure 11). Since participants had four practice runs with the *Keyboard*, we let them practice four times using our interface with no highlighted suggestions. After the first practice run, we allowed users to customize our automatically generated trigger layout (Figure 12). We used the row layout by default, but they were free to switch to the fan layout if they desired. We used these four practices with our interface to train our predictive triggering model. We then asked participants to practice the same script with all four of the interface conditions, to familiarize themselves with the setup and (in the case of *1Sugg* and *3Sugg*) the appearance of the suggestions. Since the other interface conditions are similar (or in the case of *NoSugg*, identical) to the training setup, they only practice once on those interfaces. We did not update our predictive model based on these additional practices.

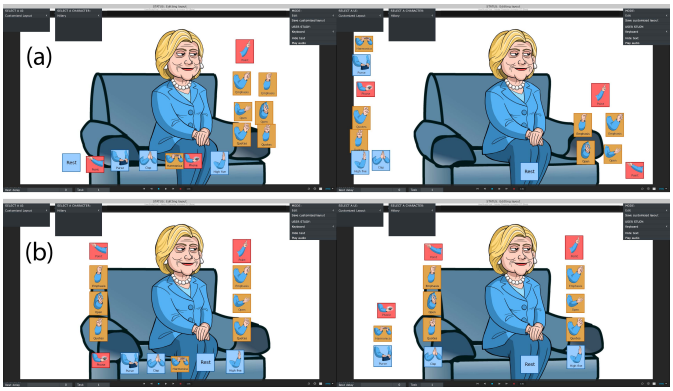


**Figure 12: Some of the customized layouts created by users during our final user study. (a) Users place the commonly used triggers near their dominate (right) hand and the unused triggers either on the bottom or off to the side. (b) Users only made minor changes to the layout.**

***Performance.*** After the practice period, participants performed the four tasks in order with the assignment of interfaces determined by the Latin square. At the end of each study session, we conducted an exit interview where participants commented on the different interfaces and ranked the tasks from easiest to hardest. Each session lasted roughly 30 minutes.

### *Findings*

We summarize the key findings from our study based on the collected rankings and comments, as well as our own observations of how participants used the various interfaces.

***Layout Customization.*** All but three users made at least some customizations to the automatically generated trigger layouts (see Figure 12). However, in most cases, the edits were relatively minor, which suggests that the automatic layout provided a good starting point for refinement (Figure 12b). That said, some users did make more significant edits, like re-arranging triggers to take advantage of a dominant hand and moving unused triggers to the side or bottom of the interface (Figure 12a).

***User Rankings.*** We analyzed the collected rankings in two ways: by task and by interface. Given that each successive task involved increasing amounts of deviation from the original scripted questions, we were expecting users to rank later tasks lower (i.e., harder) than earlier ones. However, the data does not exhibit such a trend, as seen in Figure 13a. This difference may be due to learning effects counteracting the increasing task difficulty. In contrast, the per-interface rankings show a clear preference for the three versions of our interface over the baseline *Keyboard* condition (Figure 13b). A histogram of the per-interface rankings shows that a slight preference for *NoSugg*, while *3Sugg* and *1Sugg* were very evenly matched (Figure 14). Overall, across all sessions, more than half of users preferred one of the two interfaces with suggestions over *NoSugg*.

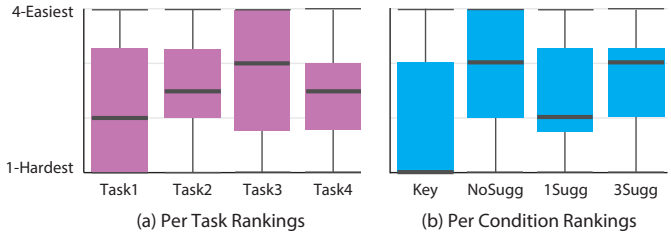


**Figure 13: The rankings from our user study organized by task (a) and by condition (b). The thick grey line is the median, the box represents 50% of the data between the first and third quartiles, and the whiskers mark the extremes.**

***Qualitative Feedback.*** The comments from the exit interviews mostly align with the rankings. Participants strongly disliked the *Keyboard* interface and complained that there was too much multitasking between reading the script on the UI, finding the correct key and then pressing it. Regarding our triggering interface, users said that the highlighted suggestions were useful and that *3Sugg* helped add variety to their performance. On the other hand, several users said that *1Sugg* was less distracting because it offered fewer options, which meant they did not need to think as much during the performance. The main criticism of the suggestions is that they made it harder to search for the intended trigger when the wrong triggers were highlighted. While a few users did voice this negative feedback, we believe this critique may be due in part to users deciding to improvise more during their performances.

### Professional Feedback

To better understand how our system addresses professional workflows, we conducted an informal demo session with the production team at The Late Show. We asked them to use our system to animate Cartoon Trump answering two questions from previous episodes of the show. In this case, one person voiced the character's responses while another person used our system to perform the triggers. They practiced three times and then performed twice, once with the *1Sugg* version of our system, and once with *3Sugg*.

The feedback was generally very positive. They liked that our system automatically generates default trigger layouts based on the artwork since they often add new triggerable poses to characters for specific shows. They also really appreciated the fact that the layouts can be customized, which allows different performers to refine trigger arrangements based on their preferences. Regarding our predictive triggering model, they

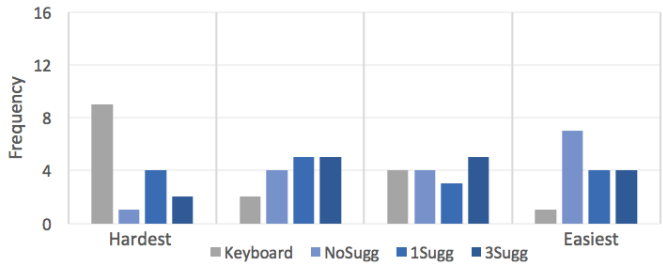


**Figure 14: A histogram showing the rankings from our user study.**

felt that suggestions for an entire performance would rarely be useful for their specific usage scenario, where the script for any given segment is only finalized shortly before the show starts. As a result, even though they do practice portions of the performance ahead of time, they typically do not have time to rehearse the entire segment before they go live. However, they noted that our suggestions would be useful for triggering the recurring sequences of poses that their characters often perform. To support this functionality, our model could automatically identify such patterns and only enable suggestions when the current triggering context matches the starting conditions for the pattern with high confidence.

| Name | Trigger Type | | | Pose Type | | | Total |
|---|---|---|---|---|---|---|---|
| | Single | Num | Radial | Hand | Face | Other | |
| Trump | 2 | 0 | 12 | 13 | 1 | 0 | 14 |
| Hillary | 13 | 3 | 7 | 13 | 0 | 0 | 13 |
| Regan | 17 | 0 | 0 | 0 | 12 | 5 | 17 |
| Furiosa | 8 | 1 | 5 | 9 | 3 | 2 | 14 |
| Cyberguy | 19 | 0 | 0 | 9 | 4 | 6 | 19 |

**Table 1: Summary of trigger and pose types for characters.**

## EXAMPLES

We used our system to animate all of the footage in our submission video and short monologues lasting 30 seconds to one minute with five characters (Figure 15). The Trump and Hillary characters were created by The Late Show production team, Regan and Furiosa were created by the author of the Red Monster Live chat session on YouTube [32], and Cyberguy was created by a multimedia artist/animator. The Trump, Hillary and Regan characters have been used previously for live animation broadcasts. All of these characters were created independently from our project, to be used during actual live animation performances. As a result, the artists have only supplied the character with the necessary poses for their particular performances. The characters have between 14 and 19 triggers, spanning the three types (single, numerical and radial) supported by our interface. While most of these triggers enable simple swaps, several characters include cycles. For example, all of Hillary's 13 hand poses are cycles that smoothly transition from the rest pose, and Regan has several triggerable cycle animations, like breathing fire and spinning eyes. In addition, Trump and Furiosa both include sub-poses that can be triggered via radial selection (e.g., moving Trump's fist up, down, left or right) or numerical selection (e.g., Furiosa holding out different numbers of fingers). Furiosa also has radial triggers that are mapped to cycles (e.g., punching a hand above the head). The animator can control the speed of the animation by moving her hand around the trigger. Table 1 summarizes the set of triggers for each character.

Our automatic trigger layout algorithm generates effective default arrangements for all of our characters (Figure 15). When creating our animated results, we only modified the layout for Regan in order to group the poses into semantically similar categories of eye triggers, eyebrow triggers, mouth triggers and others. We used automatic layouts for all the other characters.

**Figure 15: Row layouts and frames of our characters from our performed results. From left to right: Trump, Regan, Cyberguy, Furiosa and Hillary)**

To create animated monologues, we trained our predictive model on five practice runs and then performed once using the *3Sugg* mode. Assisted triggering was most helpful for performances longer than 30 seconds, where it was often hard to remember which triggers to hit later in the performance. In particular, while some poses clearly match specific verbal cues (e.g., pointing one finger while saying "first"), other poses have a less obvious connection to the speech (e.g., raising eyebrows can accompany many spoken words). In such scenarios, the highlighted suggestions provide useful prompts. The predictive model also handled mistakes gracefully. When we hit the wrong trigger, the system recovered quickly, providing useful suggestions after just one or two subsequent trigger events. Please see our submission video and supplemental web page for example sessions using all of our characters.

## CONCLUSION

Our work presents a multi-touch interface that supports triggering for live 2D animation. By generating trigger layouts based on artwork, we make it easier for users to prepare new (or modified) characters for live performance. Moreover, our predictive triggering model allows animators to leverage practice sessions at performance time.

While our results are already encouraging, we see several areas for future work.

**Additional Trigger Types.** In addition to the functionality that we currently support, some participants in our study suggested other trigger types. For example, it may be useful to have a trigger that enables continuous deformation of the artwork via direct manipulation. With this functionality, the animator could change an open hand to a closed fist and then smoothly drag the artwork to perform a punch or raise it above the character's head. Another useful feature would be a trigger that activates multiple other triggers. For instance, if a character often raises their hands, smiles and opens their eyes wide simultaneously, the animator could define a combination trigger that performs all three actions. In general, adding more triggers would help expand the expressive range of the system.

**Audio Context for Predictions.** Almost all live animations involve vocal performances. One interesting direction to explore is how to leverage context from spoken dialog to improve the predictive triggering model. For example, we could incorporate audio features captured during practice sessions into our model. Then, during a performance, the audio could serve as an additional cue to refine the suggested triggers. In this vein, we could even try to automate the triggering entirely based only on the animator's vocal performance.

**Handling Diverse Training Data.** As noted in the evaluation of our predictive model and our user study, the quality of suggestions decreases when the performance deviates from the practice sessions. One way to improve the utility of our system in such scenarios is to develop a more sophisticated model that can better handle diverse training data. In addition, the model could potentially adapt on-the-fly during a performance based on a user's actions.

In conclusion, we believe that live performed animation represents an interesting new application domain for the HCI research community. We hope that our work inspires others to investigate the unique challenges and opportunities that arise from this emerging medium.

## ACKNOWLEDGEMENTS

We thank The Late Show for helpful feedback and the use of their characters. This research was funded in part by Adobe.

## REFERENCES

1. Adobe. Character Animator. **`https://helpx.adobe.com/after-effects/character-animator.html`**, 2016. Accessed: 2016-04-10.

2. Barnes, C., Jacobs, D. E., Sanders, J., Goldman, D. B., Rusinkiewicz, S., Finkelstein, A., and Agrawala, M. Video Puppetry: A performative interface for cutout animation. *ACM Transactions on Graphics (Proc. SIGGRAPH ASIA) 27*, 5 (Dec. 2008).

3. Bell, B., Feiner, S., and Höllerer, T. View management for virtual and augmented reality. In *Proceedings of the 14th Annual ACM Symposium on User Interface Software and Technology*, UIST '01, ACM (New York, NY, USA, 2001), 101–110.
4. Christensen, J., Marks, J., and Shieber, S. An empirical study of algorithms for point-feature label placement. *ACM Trans. Graph. 14*, 3 (1995), 203–232.
5. Davis, R. C., Colwell, B., and Landay, J. A. K-sketch: A 'kinetic' sketch pad for novice animators. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '08, ACM (New York, NY, USA, 2008), 413–422.
6. Davison, B. D., and Hirsh, H. Predicting sequences of user actions. In *Notes of the AAAI/ICML 1998 Workshop on Predicting the Future: AI Approaches to Time-Series Analysis* (1998), 5–12.
7. Dontcheva, M., Yngve, G., and Popović, Z. Layered acting for character animation. In *ACM Transactions on Graphics (TOG)*, vol. 22, ACM (2003), 409–416.
8. Failes, I. How the Simpsons used Adobe Character Animator to create a live episode. *Cartoon Brew* (may 2016).
9. Findlater, L., and McGrenere, J. A comparison of static, adaptive, and adaptable menus. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '04, ACM (New York, NY, USA, 2004), 89–96.
10. Findlater, L., and Wobbrock, J. Personalized input: Improving ten-finger touchscreen typing through automatic adaptation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, ACM (New York, NY, USA, 2012), 815–824.
11. Gajos, K. Z., Everitt, K., Tan, D. S., Czerwinski, M., and Weld, D. S. Predictability and accuracy in adaptive user interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM (2008), 1271–1274.
12. Gallina, M. Cartoon Donald Trump delights audiences on the late show with Stephen Colbert. *Adobe Creative Cloud* (sept 2016).
13. Guay, M., Ronfard, R., Gleicher, M., and Cani, M.-P. Space-time sketching of character animation. *ACM Trans. Graph. 34*, 4 (July 2015), 118:1–118:10.
14. Hook, J. D. *Interaction Design for Live Performance*. PhD thesis, Newcastle University, 2013.
15. Igarashi, T., Moscovich, T., and Hughes, J. F. As-rigid-as-possible shape manipulation. In *ACM SIGGRAPH 2005 Papers*, SIGGRAPH '05, ACM (New York, NY, USA, 2005), 1134–1141.
16. Jacobson, A., Panozzo, D., Glauser, O., Pradalier, C., Hilliges, O., and Sorkine-Hornung, O. Tangible and modular input device for character articulation. *ACM Trans. Graph. 33*, 4 (2014), 82:1–82:12.
17. Kazi, R. H., Chevalier, F., Grossman, T., Zhao, S., and Fitzmaurice, G. Draco: Bringing life to illustrations with kinetic textures. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '14, ACM (New York, NY, USA, 2014), 351–360.
18. Kurtenbach, G., and Buxton, W. User learning and performance with marking menus. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, ACM (1994), 258–264.
19. Laursen, L. F., Goto, M., and Igarashi, T. A multi-touch dj interface with remote audience feedback. In *Proceedings of the 22Nd ACM International Conference on Multimedia*, MM '14, ACM (New York, NY, USA, 2014), 1225–1228.
20. Lepinski, G. J., Grossman, T., and Fitzmaurice, G. The design and evaluation of multitouch marking menus. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM (2010), 2233–2242.
21. Marin, G., Dominio, F., and Zanuttigh, P. Hand gesture recognition with leap motion and kinect devices. In *Image Processing (ICIP), 2014 IEEE International Conference on*, IEEE (2014), 1565–1569.
22. Matejka, J., Li, W., Grossman, T., and Fitzmaurice, G. Communitycommands: command recommendations for software applications. In *Proceedings of the 22nd annual ACM symposium on User interface software and technology*, ACM (2009), 193–202.
23. Messmer, S., Fleischmann, S., and Sorkine-Hornung, O. Animato: 2D shape deformation and animation on mobile devices. In *Proceedings of ACM SIGGRAPH ASIA Symposium on Mobile Graphics and Interactive Applications* (December 2016).
24. Molchanov, P., Gupta, S., Kim, K., and Kautz, J. Hand gesture recognition with 3d convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops* (2015), 1–7.
25. Momeni, A., and Rispoli, Z. Dranimate: Rapid real-time gestural rigging and control of animation. In *Adjunct Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*, ACM (2015), 61–62.
26. Motion, L. Leap motion controller. *URl: https://www. leapmotion. com* (2015).
27. Munkres, J. Algorithms for the assignment and transportation problems. *Journal of the society for industrial and applied mathematics 5*, 1 (1957), 32–38.
28. Nitsche, M., and Nayak, S. Cell phone puppets: turning mobile phones into performing objects. In *International Conference on Entertainment Computing*, Springer (2012), 363–372.
29. Oberweger, M., Wohlhart, P., and Lepetit, V. Hands deep in deep learning for hand pose estimation. *arXiv preprint arXiv:1502.06807* (2015).
30. Ponsard, A., Ardekani, K., Zhang, K., Ren, F., Negulescu, M., and McGrenere, J. Twist and pulse: Ephemeral adaptation to improve icon selection on smartphones. In *Proceedings of the 41st Graphics Interface Conference*, GI '15, Canadian Information Processing Society (Toronto, Ont., Canada, Canada, 2015), 219–222.
31. Reallusion. Crazy talk animator. **`https://www.reallusion.com/crazytalk-animator/`**, 2016. Accessed: 2016-10-10.
32. Samurai, O. Red monster live! **`https://www.youtube.com/watch?v=qBUOKI22hzI&feature=youtu.be`**, 2017. Accessed: 2017-03-20.
33. SANDERS, J. W. Disney xd drops animated livestreams for star vs. the forces of evil. **`http://promaxbda.org/brief/content/disney-xd-drops-animated-livestreams-for-star-vs.-the-forces-of-evil`**, 2017. Accessed: 2017-03-20.
34. Scribbleh. Scribbleh. **`https://www.twitch.tv/scribbleh`**, 2016.
35. Snap. Snapchat lenses, 2016.
36. Sturman, D. J. Computer puppetry. *IEEE Comput. Graph. Appl. 18*, 1 (Jan. 1998), 38–45.
37. Supancic, J. S., Rogez, G., Yang, Y., Shotton, J., and Ramanan, D. Depth-based hand pose estimation: data, methods, and challenges. In *Proceedings of the IEEE international conference on computer vision* (2015), 1868–1876.
38. ToonBoom. Harmony: Animation and storyboarding software. **`https://www.toonboom.com/`**, 2016. Accessed: 2016-04-10.
39. Vollick, I., Vogel, D., Agrawala, M., and Hertzmann, A. Specifying label layout style by example. In *Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology*, UIST '07, ACM (New York, NY, USA, 2007), 221–230.
40. Wei, S.-E., Ramakrishna, V., Kanade, T., and Sheikh, Y. Convolutional pose machines. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2016), 4724–4732.
41. Williams, E. Predictive, adaptive mobile user interfaces: State of the art and open problems. In *Proceedings of the 2014 ACM Southeast Regional Conference*, ACM SE '14, ACM (New York, NY, USA, 2014), 35:1–35:3.
42. Williams, L. Performance-driven facial animation. In *ACM SIGGRAPH Computer Graphics*, vol. 24, ACM (1990), 235–242.
43. Yu, X., Yang, J., Luo, L., Li, W., Brandt, J., and Metaxas, D. Customized expression recognition for performance-driven cutout character animation. In *Proceedings of Winter Conference on Applications of Computer Vision (WACV)*, IEEE (2016).
44. Zhai, S., Hunter, M., and Smith, B. A. Performance optimization of virtual keyboards. *Human-Computer Interaction 17*, 2–3 (2002).