SketchExpress: Remixing Animations for More Effective Crowd-Powered Prototyping of Interactive Interfaces

Sang Won Lee, Yujin Zhang, Isabelle Wong, Yiwei Yang, Stephanie D. O'Keefe, Walter S. Lasecki

Computer Science & Engineering | University of Michigan
{snaglee,yujinz,iswong,yanyiwei,sdokeefe,wlasecki}@umich.edu

ABSTRACT
Low-fidelity prototyping at the early stages of user interface (UI) design can help designers and system builders quickly explore their ideas. However, interactive behaviors in such prototypes are often replaced by textual descriptions because it usually takes even professionals hours or days to create animated interactive elements due to the complexity of creating them. In this paper, we introduce SketchExpress, a crowd-powered prototyping tool that enables crowd workers to create reusable interactive behaviors easily and accurately. With the system, a requester—designers or end-users—describes aloud how an interface should behave and crowd workers make the sketched prototype interactive within minutes using a *demonstrate-remix-replay* approach. These behaviors are manually demonstrated, refined using remix functions, and then can be replayed later. The recorded behaviors persist for future reuse to help users communicate with the animated prototype. We conducted a study with crowd workers recruited from Mechanical Turk, which demonstrated that workers could create animations using SketchExpress in 2.9 minutes on average with 27% gain in the quality of animations compared to the baseline condition of manual demonstration.

ACM Classification Keywords
D.2.2. Design Tools and Techniques: User Interface; H.5.m. Information Interfaces and Presentation (e.g. HCI): Misc.

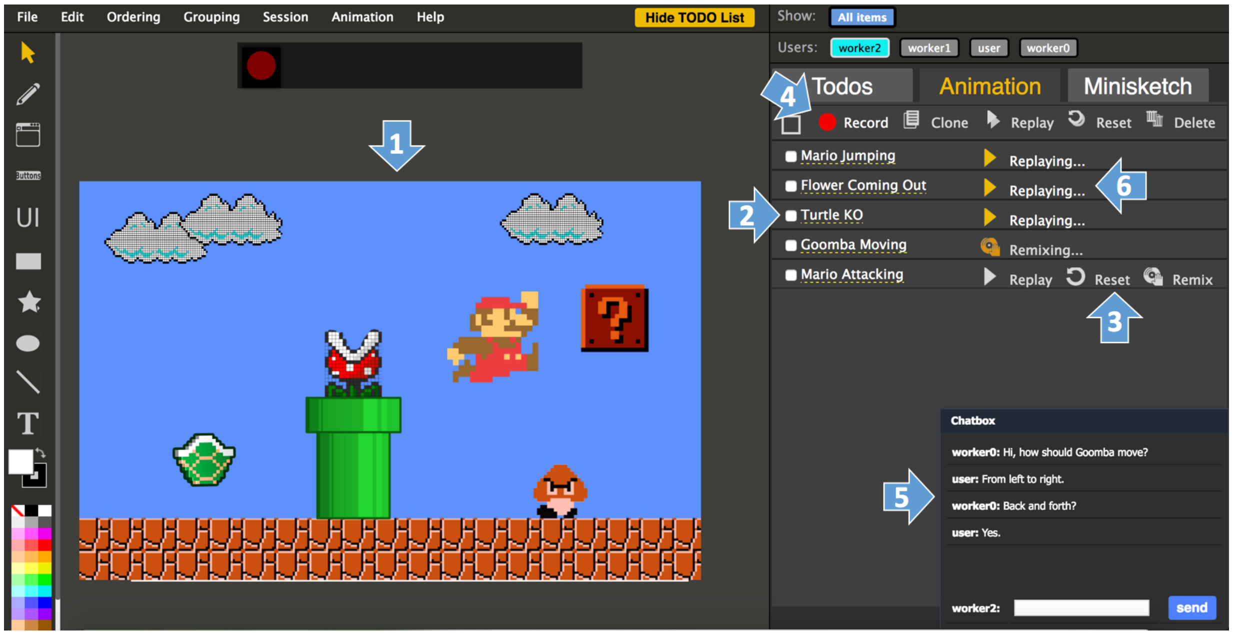
Author Keywords
Rapid prototyping; Animation; Real-Time Crowdsourcing; Human Computation; Interactive interfaces;

INTRODUCTION
Prototyping graphical user interfaces (GUIs) allows designers to clearly envision the effect that their design decisions will have in practice and enables rapid, informed iteration. Designing interactive behaviors often involves creating low-fidelity prototypes that can actually be *used* as opposed to a static drawings, which are not interactive.

However, creating interactive behaviors in early stage prototypes is challenging for multiple reasons. First, interactive behaviors involve the dynamic transformation of multiple interface elements, which indicates that they cannot be easily presented in static images. For example, to demonstrate how a user can "swipe to unlock" a smartphone screen, a static sketch is not enough. It requires a description of cause and effect behaviors, e.g. what happens to the button with the arrow moving within the rail when it reaches the right end, what happens when a user releases the button halfway.

Second, existing tools typically provide a set of predetermined behaviors that one can choose from. These preset behaviors tend to only support specific types of applications (e.g., transitions between web pages, or drop-down widgets) well. As a result, these tools are limited to prototype behaviors in general systems that go beyond traditional window-based GUIs. For example, animating the behaviors of a video game character in a side-scrolling game (e.g., Super Mario) or how the enemy characters (e.g., turtles in Super Mario) respond to the other element's changes (e.g., Mario bouncing on them) cannot be accomplished easily with existing tools. Existing tools that can support expressive interactive behaviors require expertise – and even then, creating high-fidelity prototypes can take hours or days even for professional designers, which makes them inappropriate for use in the earliest stages of UI design. Interactive prototyping requires first learning these professional tools, making it difficult for non-experts (i.e. UI end users) to participate in the UI design process.

In this paper, we build upon Apparition [17], a system that leverages the online crowd to create interactive behaviors for the designer. Apparition is a crowd-powered prototyping tool in which a requester describes a GUI aloud and through sketch while crowd workers connected to a shared canvas update and refine the prototype. In Apparition, crowd workers can demonstrate behaviors by manually animating objects whenever a requester interacted with the sketch. However, the complexity of behaviors that crowd workers could demonstrate was limited. In addition, the manually demonstrated behaviors are ephemeral and do not persist with the sketch once the sketching session is over. While we believe this model of real-time collaboration between a requester and crowd is powerful—as it allows end users to quickly and naturally create GUI prototypes—it necessitates an effective way for non-expert crowd workers to create interactive UI behaviors on the fly.



worker0: Hi, how should Goomba move?

user: From left to right.

worker0: Back and forth?

user: Yes.

Figure 1. SketchExpress allows crowd workers to prototype interactive behaviors. A requester describes aloud how a user interface should behave and crowd workers quickly create complex interactive behaviors. The interface contains the following features for crowd workers to easily create interactive behaviors (animations) as follows: *(1)* A synchronized canvas that supports simultaneous interactions between a requester and workers. *(2)* the ability to select and replay multiple animations at once; *(3)* reset functionality that places elements in the animation back to their initial state (position, color, etc.); *(4)* recording button to record a worker's demonstrations; *(5)* a chat box for helpers to ask clarification questions if needed. *(6)* labels that show the current state of the animation [replaying/remixing] to prevent multiple workers from concurrently working on the same animation.

This paper build upon this crowd-powered prototyping model by providing techniques for crowd workers to easily and accurately create interactive behaviors with a rich range of expression. To do this, we introduce SketchExpress, a tool that is accessible to non-expert designers, that allows crowd workers to create higher fidelity, *reusable* animations within a few minutes in order to power interactive prototypes.

SketchExpress is built on top of Apparition, and the envisioned interaction between the requester and the workers is drawn from that system [17]. In SketchExpress, designers—*requesters* in this crowdsourcing context—verbally describe their prototype and one or more crowd workers collectively produce a corresponding sketch. A requester can draw content and then describe aloud desired behaviors without having to stop to implement/create the functionality they are describing. Behind the scenes, non-expert crowd workers listen to the verbal requests and use SketchExpress's UI to create replayable animations in a matter of minutes. Designers can also mock-up interactive behaviors using SketchExpress without the crowd, but crowdsourcing allow the system to make the creation process fluid and quick, which in turn makes requesters' interaction with the system minimal and natural.

We introduce an effective way for crowd workers to create interactive behaviors quickly and accurately in response to designers' requests. SketchExpress does this by introducing the *demonstrate-remix-replay* method, which is easy to learn and expressive enough to prototype complex behaviors. Based on the requester's verbal description, crowd workers first *demonstrate* a behavior that is recorded by the system as a series of operations. Workers can then *remix* the recorded animation to further refine it. Once this is done, any worker or designer can *replay* and compose multiple animations with a click of a button, making it possible to effectively support complex (multi-part) animations in early-stage prototypes. The resulting prototype retains complex behaviors and can be used to iteratively explore design ideas, communicate with collaborators, and act as a "living spec" for future implementation.

We make the following contributions in this paper:

- a novel method, *demonstrate-remix-replay*, with which non-expert crowds can prototype interactive GUI behaviors;
- SketchExpress, a system that creates reusable, higher-fidelity animations in early sketch;
- validation of our approach through a user study with crowd workers recruited from Mechanical Turk.

The results in this paper contribute to the broader goal of creating a prototyping tool that helps anyone design and/or modify GUIs. We aim to support a broad population: designers to rapidly iterate on ideas and hand-off tasks to crowd workers: non-experts to participate in the design and improvement of software systems; researchers to quickly mock up interactive tools for experimentation; and students to create engaging examples even before they learn to program. Within the scope of this paper, we focus specifically on techniques crowd workers can use to create interactive behaviors in prototypes.

We begin by reviewing the challenges and related work in prototyping interactive behaviors and crowdsourcing. We then discuss what makes interactive behaviors complex to manually demonstrate, and how this guides the design choices we made in SketchExpress. Finally, we report the results of a user study that we conducted with real crowd workers.

BACKGROUND AND RELATED WORK

SketchExpress builds upon prior work on: 1) prototyping tools that make dynamic and interactive sketches; 2) continuous real-time crowdsourcing; and 3) end-user programming by demonstration systems. We discuss prior work in these domains to provide context for the design choices made by SketchExpress to help make early stage prototyping of interactive behaviors more accessible to both non-expert designers (requesters) and workers. We also review existing tools that permits users to create interactive behaviors in GUI prototypes.

Designing Interactive Behaviors in Sketching Tools

UI prototypes are used by system builders to explore new ideas in depth more quickly. Rather than building fully functional systems from the beginning, prototypes permit quick trial and error iterations that can be easily produced and evaluated. Systems like SILK [16] and DENIM [23] were early efforts that reduced the overhead of prototyping by recognizing designers' sketches as interface elements and implementing the idea of wireframing, respectively. However, the outcome of such tools is most often a static sketch that does not include the interactive aspects of the GUI. Designing UI behaviors is harder than designing layouts because the behaviors are more complex to demonstrate and the tools available to designers have more limitations [27].There are several professional UI prototyping tools that can be used to program interactive behaviors, and though these tools have become more user-friendly, they are still difficult and time consuming for non-experts to learn and use. Additionally, these tools often support only a limited set of animations in specific UI contexts (e.g. wireframe transitions, standard widgets for mobile applications), which makes it difficult to prototype interactive behaviors for general applications. We will discuss existing tools more later in this section. SketchExpress makes the creation process natural and expressive, recruiting crowd workers "power" interactive sketches using the demonstrate-and-remix approach without spending extensive time learning how to use complex tools.

Previous research has also created tools that support dynamic sketches and are easy to learn. For example, non-expert users were able to learn K-sketch within 30 minutes and use it to generate dynamic illustrations that can be played as an animation within 7 minutes [8]. SketchExpress draws on a number of important ideas about recording demonstrations, dubbing, and post-edits from K-sketch and other similar systems [2, 33]. One important distinction between SketchExpress and K-sketch is that K-sketch creates a single, linear series of actions to represent a behavior (as if it were a video), whereas SketchExpress generates a set of animations that can be replayed independently and simultaneous, allowing workers to mix and match existing actions to represent new behaviors. Thus, SketchExpress prototypes can end up in various states depending on which combination of animations are executed.

Alternatively, Sketchify lets designers generate completed interactive behaviors through a scripting language [32]. Users can write scripts to configure subtle relationships between elements and interactive materials (e.g., sensors), focusing mainly on the interactivity and integration with other input sources. Their study showed that scripting "does not fit" the overall sketching system and it also confirms our belief that too many functions "may cause confusion and overload" [32]. SketchExpress transforms a static sketch into an animated prototype without programming and leverages human computation to handle aspects that would otherwise require script logic.

More recently, Kitty employed various methods to enable a dynamic relationship between elements on canvas [13]. While Kitty was developed for artists to create illustrative animations, SketchExpress utilizes a more traditional sketching tool. The process of creating animations in Kitty is close to programming, using: kinetic textures, relational graphs, and functional mappings. However, to crowdsource prototyping we need much simpler yet similarly expressive interaction techniques that non-expert workers can pick up nearly instantaneously.

Most existing tools attempt to simplify the programming process of interactive behaviors, retaining the logic behind the behavior. In contrast, SketchExpress records manual demonstration and lets crowd workers remix it to refine the behaviors. This Wizard-of-Oz approach has been shown effective in making the design process accessible to a broader population [9, 26], but has been used on static UI sketches rather than interactive components. An adaptation of the Wizard-of-Oz approach where human operators manipulate paper prototypes to show interactive components has been traditionally used to demonstrate the dynamic behaviors of prototypes [11]. Animating physical mockups is a widely used and powerful technique, but it is limited by the physical efforts needed to produce the cutouts and to manually animate the objects. For example, the materiality of physical mock-ups makes some types of behaviors difficult to demonstrate (e.g. scaling, re-coloring, opacity). In addition, using videotaping and editing to replay demonstrations of physical mock-ups can be used, but again yields a single linear progression of actions, in contrast to SketchExpress's recomposable animations that result in greater expressivity. Lastly, SketchExpress has the advantages of electronic sketching (discussed in detail in [16]). For example, an electronic sketch can be quickly drawn, easily modified using operations (i.e., *save*, *copy*, *paste*, and *edit*), and shared with others (e.g., for remote collaboration).

Crowdsourcing and Human Computation

Crowdsourcing for human computation engages people through an open call to contribute to a computational process in order to solve problems that machines cannot solve alone. Crowd workers can be recruited on demand from platforms like Amazon Mechanical Turk [1]. These workers are often quasi-anonymous to requesters [20], and have unknown and varied skills/experiences. In crowdsourcing, work is often coordinated by dividing it into small, context-free units called "microtasks" [24]. Significant effort is required to generate microtasks and aggregate responses. While microtasks have been shown to be useful for tasks that have a clear goal and an established problem-solving process, open-ended tasks (e.g., designing a UI) are better solved through collaboration between requesters and workers [6].

Continuous real-time crowdsourcing [18] can address this challenge since it not only elicits rapid responses but also enables requester-worker interactions [19]. VizWiz [5] showed that the crowd could answer visual questions in under a minute. Legion [18] introduced continuous real-time crowdsourcing for collective control tasks, and Adrenaline [3] used a "retainer model" to bring and direct crowds to a task in seconds. Legion-Tools [10] builds on this idea, and is the first publicly-released tool for recruiting and managing real-time crowds.

However, tasks that do not have a fixed process and require continuous involvement, like designing a UI prototype, have been under-studied. An early crowd-powered system that handled open-ended tasks was Soylent, which provided document-editing assistance, such as proofreading or text shortening with a general process: Find-Fix-Verify [4]. Research has continued to find new ways to coordinate workers on writing tasks for academic writing [31], creative writing [34, 14], and on community resources like Wikipedia [15].

Programming by [Remixing] Demonstration

Defining interactive behaviors by first demonstrating and later remixing them is a response to the trade-off between the expressiveness of resulting behaviors and the sophistication of the creation process [29]. This is a simple form of End User Programming (EUP) [30], and as such, faces similar challenges in making the flexibility of computation accessible to non-experts (for an overview, see: [28]). We draw ideas from Programming by Demonstration (PbD) [7], which explores how a user's manual demonstration can specify a program – or interactive behaviors, in our case. To address this challenge, we incorporate the notion of "remixing", an idea commonly used in electronic music: i.e., a DJ chopping, editing, processing, and arranging audio samples to create music. The idea of remixing to facilitate real-time collaboration draws upon the previous work in collaborative improvisation, where musicians can algorithmically remix short musical patterns [22] or musical notation [21].

Summary of Tools for Prototyping Interactive Behaviors

A range of alternatives exists for creating realistic interactive behaviors. We have reviewed fifteen such tools to evaluate existing approaches, examining how the tools support a variety of interactive behaviors. Overall, we identify the following three categories of tools.

1. [high programmability - rich expressivity]: these applications (e.g., Kitty, Sketchify, Flinto, Origami) provide methods with which a user can specify relationships and states between objects, equivalent to a programming environment, which yields interactive and dynamic sketches. This class of applications often comes with numerous complex configurations that tend to be time-consuming to learn and understand in order to harness the applications' full range of expressivity.

2. [preset behaviors for target applications]: these widely used prototype tools (e.g., InVision or Adobe Experience Design) provide a limited library of behaviors that a user can choose from for typical common applications (e.g., page transitions, image overlays, and hyperlinks). However, users may struggle both with complex configuration, which is proportional to the number of prepared behaviors, and limitations of the existing preset if the desired behavior is not one of the predefined ones. Hence, this class of solutions cannot handle general applications, such as games or animated illustration.

3. [linear timeline]: these applications (e.g., Atomic.io, Adobe After Effects, K-Sketch) provide a linear timeline editor that is typically available in film-editing software. While this class of applications offers rich expression—as a designer can manipulate elements over time-dimension (frame by frame)—the linearity of the animation limits the dynamics and interactivity of the prototype. Typically, one behavior can be expressed linearly, but a GUI prototype that has a number of behaviors cannot be created using a single timeline-based animation because it can require different outcomes depending on how a user interacts with it. For instance, pawns in a chess game have limited behaviors, but the number of possible states that a chess game can end up in is extremely large.

Most of these applications provide programming-like functionality (or something equivalent) for prototyping interactive behaviors. However, generating expressive animations often comes at the price of learning the tools in depth and attaining expertise in at least one of the required programming concepts. This can be a barrier when utilizing crowd workers to prototype interactive behaviors. We reduce the effort needed to create interactive behaviors via the demonstrate-and-remix-replay approach. Reducing the effort needed to create animations provides allows requesters to explore interactive behaviors more quickly. Rapid prototyping and iteration is particularly important as the system targets the early stages of the design process, where people often exchange ideas by drawing on a piece of paper (or so-called "napkin sketch").

THE COMPLEXITY OF INTERACTIVE BEHAVIORS

The complexity of interactive behaviors can vary highly. They can be as simple as menu items expanding when a drop-down menu is clicked, or as complex as a game character throwing fireballs to defeat an enemy character. The dynamic, state-dependent nature of interactive behaviors makes creating and automating them difficult without the use of programming (or something equivalent, such as timeline editors, parameters configuration, or visual programming).

Why Programming? Typically, interactive behaviors have three properties: $P1$: what *triggers* the interactive behavior, $P2$: the (visual) state changes made to the UI, and $P3$: the effects of the interactive behavior on the underlying state, which may affect following behaviors. For instance, in our Super Mario example, the textual description of an interactive behavior can be: "When Mario jumps and lands on top of an enemy turtle ($P1$), the turtle should hide its legs and head in the shell and stop moving ($P2$); the shell can then be used as a weapon if Super Mario pushes it ($P3$)."[1] The description includes user interaction, state recognition, trigger conditions, animated behaviors, and abstract state changes. In addition to animated visual changes, abstracted conditions that influence triggering animations (here, $P1$ and $P3$) make it difficult to realize the behavior without programming.

[1]https://youtu.be/rLl9XBg7wSs?t=17m38s

Why Wizard-of-Oz? SketchExpress uses a collective Wizard-of-Oz approach to address $P1$ and $P3$, letting crowd workers manually demonstrate interactive behaviors for visual changes ($P2$). It is relatively easy for a crowd "Wizard" to understand the triggering condition ($P1$) and relationships between elements ($P3$). In [17], the authors showed that crowd workers can accurately respond to user interactions in seconds and vary the demonstration based on the system state. However, it can be challenging for a worker to demonstrate complex animated behaviors due to the limitations of manual demonstration (i.e., limits in human motor abilities, accuracy, and the number of simultaneous actions). Thus, we focus on improving the animation process ($P2$) and study how we can computationally support workers when creating complex interactive behaviors composed of multiple animations, which can be challenging, or even impossible, to manually demonstrate.

Limitations of Manual Demonstration

In this section, we outline the key challenges in manually demonstrating interactive behaviors accurately.

Temporal Execution of Interactive Behaviors

A complex interactive behavior includes a sequence of different kinds of state changes. We define an *operation* as one or more events of the same kind enabled by one user interaction (such as a click, drag-and-drop, or pressing of a key) where an *event* is an atomic state change in canvas (e.g., color change, translation). One operation can be a long sequence of events when the operation involves smoothly animated elements (e.g., moving, resizing, rotating). We concluded that an *operation* is the most effective building block for describing an interactive behavior and for notating the process of a manual demonstration for later remixing. Recall the knock-out gesture of the turtle in Super Mario game[2]: the turtle shell should bounce off the ground once flipped upside down. The operation of flipping upside down should happen instantly (rotation operation), and the operation of bouncing off the ground should occur for a fixed duration (translation operation). To manually demonstrate this behavior, one can rotate the turtle shell image to make it flipped upside down, and drag-and-drop the icon over a trajectory. However, not only is it difficult to drag-and-drop the icon exactly within a fixed animation duration, but it is physically impossible to instantly rotate the icon and drag-and-drop immediately after the rotation because this requires grabbing the icon in two exclusive cursor modes, which requires a mode switch. The eventual state after the manual demonstration may be correct, but the intermediate process is far from the realistic – the demonstrated behavior takes several seconds with delays between operations. In SketchExpress, the remixing functionality allows workers to adjust the duration of each step in an animation.

Simultaneous Transformation

Another challenge in certain behaviors is simultaneously transforming multiple elements in different ways. Suppose a user wants to illustrate a stone rolling down hill. The stone should be translated in one direction and rotated at the same time. This behavior cannot be demonstrated manually because these are two exclusive operations (translation and rotation). One possible solution would be to demonstrate the two different animations separately and replay them simultaneously. If multiple animations transform the same element, the resulting animation should be then additively synthesized (known as *dubbing*). To this end, we specifically implemented the record function inSketchExpress to calculate the difference between two events (the delta, Δ), instead of recording a series of snapshots for replay exactly as demonstrated.

[2]https://youtu.be/rLl9XBg7wSs?t=10m24s

Animation as a re-usable object

In many GUIs we reviewed, we found that common behaviors are often shared by different objects. However, in manual demonstration, the demonstrated behavior can only be bound to one specific element during execution. Since SketchExpress retains behaviors after demonstration, it can help make animations reusable for different elements and use them in a generative fashion (i.e., clone and concatenate them).

Advanced Visual Effects

There exist dynamic transformations that cannot be demonstrated without generative or deformative techniques. Such transformations are typically available in professional animation software (e.g., Adobe After Effects, or Autodesk Flame) For example, morphing one object into another object or visual effects (such as an explosion + particles) are difficult to manually demonstrate without simplification. We exclude these kinds of advanced transformations, as such effects are not essential in early sketches and can be simplified with images.

In this section, we reviewed what characteristics of interactive behaviors make them challenging to manually demonstrate. In the next section, we address these challenges and describe the design choices we made in developing SketchExpress.

SKETCHEXPRESS

SketchExpress's goal is to let crowd workers help create and power behaviors in interactive GUI prototypes easily and accurately. To recap, the primary challenges are:

- archiving manual demonstrations as reusable and replayable animations;
- allowing for remixing of manually-demonstrated animations to have more precise timing;
- creating interactive behaviors that animate multiple elements simultaneously;
- allowing crowd workers to learn to do the tasks above within minutes of first using the system.

In the rest of this section, we introduce the animation functions that crowd workers are able to use to prototype interactive behaviors. We then describe the implementation of SketchExpress in detail and discuss how it addresses these challenges.

Platform

SketchExpress is built on top of the existing web-based Apparition system (see [17] for more detail) which provides a shared canvas (modified from SVG Edit, a web-based SVG drawing application [25]) where a requester and a crowd worker can collaborate in real time. A requester verbally describes an interactive behavior via streaming audio channel while sketching on the canvas. The requester's interface is same as the

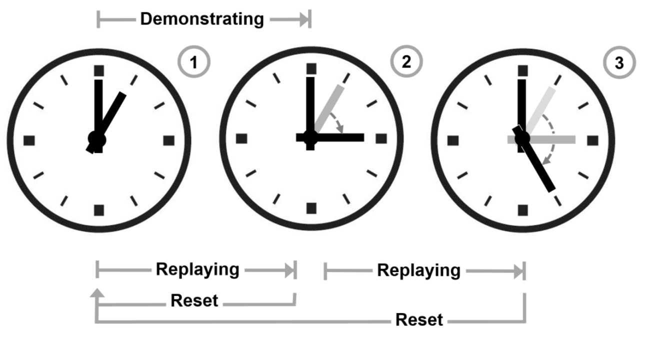


Figure 2. *(1)* Initial State: arrow pointed vertically upwards. *(2)* After first replay: arrow pointed 60 degrees clockwise. *(3)* After second replay: arrow pointed 120 degrees clockwise. Reset will restore state (1).

worker's interface except that it has a simplified tool bar with only a free-hand drawing tool (pencil) and a select tool. As the canvas is synchronized in real-time, the requester and the workers see the same content and updates.

Recording Demonstration

SketchExpress provides the ability to record manually demonstrated behaviors, which are stored as a series of time-stamped snapshots of the element that can be later replayed as an animation. To record, workers: 1. press record (Fig.1-4), 2. demonstrate the behavior on the canvas, 3. stop recording, and 4. the server post-processes the recorded log to construct a replayable animation. Each recorded animation can be *replayed*, *reset*, and *remixed*. The state of each animation is shared in real time in the side panel to provide awareness (Fig.1-1), which helps avoid conflicts in replaying and remixing animations. Pressing the *Replay* button triggers the animation to begin again. Pressing the *Reset* button restores the initial states of the elements used in the animation (Fig.1-3).

One key benefit of using the record-and-replay method compared to manual demonstration is that the interactive behaviors then persist as part of the sketch and can be replayed later. While the recording function simply logs all the snapshots of the elements that are changed by the crowd worker, Step 4 categorizes events and generates a series of operations.

There are three supported types of operations: `create`, `change`, and `delete`. The `change` operation can be broken into five sub-categories: `move`, `rotate`, `resize`, `fill-change`, and `stroke-change`. Depending on its type, one operation can have a series of multiple *events* (e.g., `move`) or a single event, typically color changes (e.g., `fill-change`, `stroke-change`). When replaying an animation, each operation is reproduced in real-time as it was demonstrated during recording, including the delays between operations. The operations are listed, showing the type of each operation and the associated timing, in a table in the remix panel (Fig.4).

Recording and Replaying By Delta

In the post-processing step (Step 4), two adjacent events are compared to compute the difference (Δ) between two snapshots within an operation. The replayed behavior is thus the relative state difference between the initial state and the ending state ($A' - A$) rather than an absolute frame-for-frame reproduction of the demonstration. There are a few advantages in terms of expressiveness to replaying an animation by delta (as opposed to via a series of snapshots).

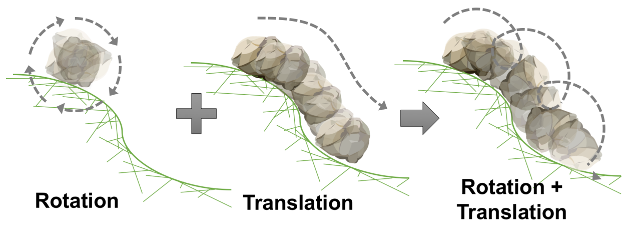


Figure 3. SketchExpress facilitates the process of creating complex animations involving multiple transformations on one element. In this example, a worker can create separate *rotate* and *translate* operations, and then replay them together to create the rolling stone animation.

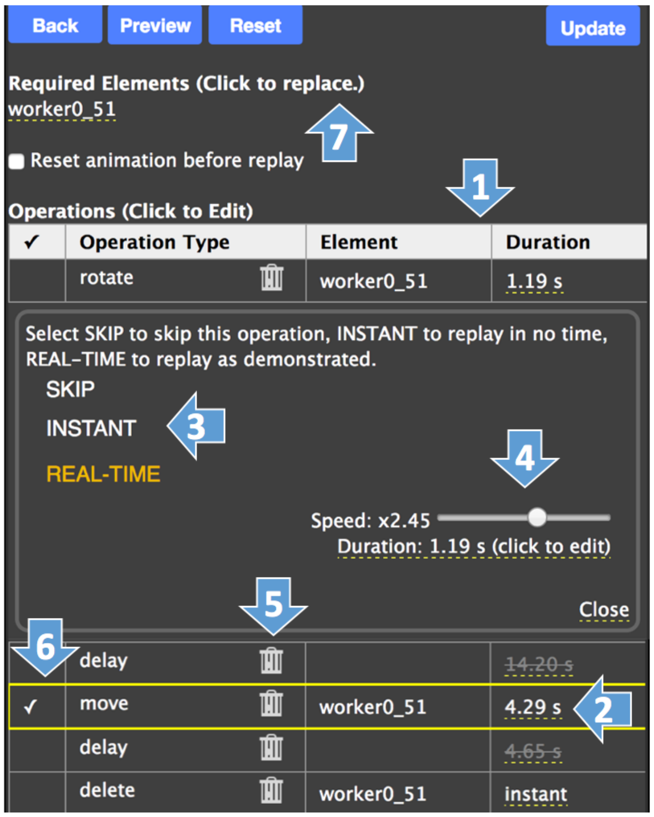
First, an element can 'own' a behavior instead of the behavior being reproduced exactly as it was demonstrated. Depending on the current state of the element, each behavior may result in a different animation and yields different outcomes. For example, if an animation is created by recording the rotation of an object by 60 degrees, the resulting animation is not an exact reproduction of the originally recorded animation but instead a rotation by 60 degrees from the element's current position (see the example in Fig.2). Second, this enables multiple transformations on one element. Automated replay of multiple animations enables complex behaviors that cannot be easily demonstrated by manipulating them (see the rolling stone example in Fig.3). A worker can replay and combine animations in any order to simulate UI behaviors.

Remixing Animations

SketchExpress provides remixing functions to control the timing of each operation of a recorded animation. While manual demonstration can be spatially expressive, the temporal execution of the demonstration is limited by the time it takes to physically animate the elements. Using remix, workers can adjust the duration of each operation in an animation.

The main interface for remixing is the remix table (Fig.4-1). For each operation, there are three options to choose from: `instant`, `skip`, and `real-time` (Fig.4-3). `Instant` makes the transition from the operation's initial state to the final state occur immediately, which is useful when the intermediate operations are not needed in the replay (e.g., for a "teleportation" animation). `Skip` allows one to bypass the recorded operation, which is useful to remove unnecessary actions captured while recording. `Real-time` replays the operation as it was demonstrated with a specified duration that can be stretched or compressed (Fig.4-4). *Real-time* exactly reproduces the recorded demonstration if the duration is not modified.

Depending on the type of animation, the initial state of each replay needs to be reset before/after replay and can be looped when it is periodic (e.g., a non-player character patrolling in a game). These options not only automate some of the process, but increase the kinds of behaviors that SketchExpress can present. Overall, remixing an animation in the temporal dimension is a key function in transforming a demonstration



Required Elements (Click to replace.)
worker0_51

Reset animation before replay

Operations (Click to Edit)

✓	Operation Type	Element	Duration
	rotate	worker0_51	1.19 s
	delay		~~14.20 s~~
✓	move	worker0_51	4.29 s
	delay		~~4.65 s~~
	delete	worker0_51	instant

Select SKIP to skip this operation, INSTANT to replay in no time, REAL-TIME to replay as demonstrated.

Speed: x2.45
Duration: 1.19 s (click to edit)

Figure 4. Remixing helps workers make more expressive animations. The interface consists of: *(1)* Operation list: provides workers with a discrete view of an animation as a series of operations. *(2)* Operation duration: if clicked, a container of remix functions is expanded. *(3)* Replay options: you can choose for each element if it will be skipped (not displayed), if it will appear instantly, or if it will appear in real-time. *(4)* Slider and input: modify the operation's speed and duration. *(5)* Trash icon: skip the operation or delay. *(6)* Check mark and highlight border: indicate the current operation in preview. *(7)* Element replacement: reuses the animation for one element as the animation for another one.

into a precisely timed animation, which not only makes the animation look smoother, but also allows workers to demonstrate behaviors without being concerned with making the initial demonstration perfectly temporally accurate.

One challenge for workers is to associate the contents of the remix table with the animated elements on the canvas. SketchExpress provides multiple visualization techniques for workers to connect entries with canvas elements. First, when recording an animation, the list of operations is generated on-the-fly during demonstration, allowing the worker to immediately associate actions on the canvas with entries that will later be used to remix the animation. Second, whenever an animation is replayed, the entry corresponding to the currently-playing operation is highlighted. The delay row is highlighted if the animation is in between operations. Lastly, whenever a worker places their cursor over one a row in the table, the elements associated with that operation are highlighted (as seen in programming environments that highlight program outcomes associated with code text [12, 35]). In order to clearly visualize "what is remix-able", we made use of a consistent format: a yellow-dotted line under options throughout the remix table that can be clicked and remixed.

One of the benefits of the demonstrate-remix-replay approach is that it is easy for non-experts to understand the controls that they are given. The expressiveness afforded by SketchExpress is defined by the multiplication of two orthogonal dimensions (time and space). For spatial dimension, it includes any change that a worker can make in the drawing application, which can be controlled in the *demonstrate* phase. For temporal dimension, there are four types of control actions that can be conducted per operation: `compress`, `stretch`, `skip`, and `instant`. While we could have created remix functions that can modify the spatial data of a demonstrated behavior, we deliberately limited the remixing capability only to the temporal dimension of an operation and delays in between, helping workers more quickly understand the tool's range of expressiveness. Therefore, if the demonstrated behavior is not spatially correct, workers need to re-record the behavior again, leveraging humans' fine motor function for the expressiveness given there's no time pressure. While we could have added spatial remix functions to correct visual trajectory of animation, we believe that the simple structure of demonstrate-remix helps non-expert crowd workers learn to use the tools by themselves and use them after a brief exploration.

Animation as a First Class Object

SketchExpress provides features that treat animations as independent from the elements on the canvas – akin to a first-class programming object. For example, a worker can "clone" an animation and switch the element that is used in the animation so a certain behavior can be applied to different elements (e.g., letting us apply our example turtle knock-out behavior to other enemy characters). A worker can replace existing elements of an animation in the "Required Elements" and "Created Elements" list below the buttons in the remix mode (Fig.4-7). Once switched, any operation that was associated with the original element works for the new one. To avoid orphaned animations, when a user deletes an element associated with an animation, SketchExpress alerts a worker with a list of the affected animations. Cloning an animation can be used to create multiple remixed versions of one demonstrated behavior. Finally, animations can be imported/exported across sessions by archiving them *json* content. Treating animations as first-class objects lets workers easily compose new animations.

EXPERIMENTS

User Study - UI Tasks

To verify SketchExpress' ability to help crowd workers prototype interactive behaviors in various UIs, we ran a study in which crowd workers were given behavior descriptions and asked to collectively create them using our interface. We first selected five common interfaces that require complex interactive behaviors (difficult to demonstrate) from various domains (from mobile to game design). Our study controlled for variation in natural language descriptions by having one of authors read from a script describing the tasks across the sessions. Since the goal of this work is to confirm if crowd workers can create behaviors easily and accurately, we wanted to limit the chance that confusion would arise from variations in either verbal communication or the description of task content. We carefully generated the script to reflect the target use cases by transcribing a non-designer, who is not an author on this paper, verbally describing the interactive behaviors in the tasks. As this study focuses on system feature effects on the interactive behaviors, not the static parts of the sketch, a graphical user interface is given to crowd workers and the interface has all the elements necessary for a worker to demonstrate the behaviors that will be requested. Crowd workers are instructed to create interactive behaviors for each task, resulting in a total of nine interactive behaviors across five sketches. Each task (T) and interactive behavior (IB) is described to workers as follows:

- Task 1 (**T1**): Super Mario Game – on the ground, Super Mario jumps to defeat a turtle (a.k.a. Koopa) and an enemy mushroom (a.k.a. Goomba)
  - IB1: Super Mario jumping forward
  - IB2: Turtle Knock out gesture
  - IB3: Mushroom Knock out gesture
- Task 2 (**T2**): Traffic lights Demonstration
  - IB4: Traffic light changing color from green to yellow to red with a two second delay between each change
- Task 3 (**T3**): To-do List Application
  - IB5: Crossing off an item (the 1st item in the list) by showing a check mark in a check box and a strike-through the text simultaneously and instantly
  - IB6: Crossing off an item (the 2nd item in the list)
- Task 4 (**T4**): A cannon-firing game
  - IB7: A cannonball from the pile of cannonballs is loaded into a cannon barrel, at which point the cannon shoots it out to destroy an enemy character
  - IB8: Same as IB7 for another cannonball and the second enemy character
- Task 5 (**T5**): Unlock screen
  - IB9: A user "swipes to unlock" a smartphone screen

These tasks focus on remixing animations rather than the reusability, for which benefits may emerge over time. For example, making IB3 could have benefited from re-using IB2.

Participants

We recruited 18 unique crowd workers from Mechanical Turk who have never used SketchExpress before. We limited the crowd workers to those who are in the U.S. and have an approval rate of over 70%. All workers who applied for the work were asked four binary questions to see if they were eligible to complete our user study: 1) if they can listen to verbal instructions through audio streaming, 2) if they are familiar with at least one common creative application (Microsoft PowerPoint/Microsoft Point/Google Draw/Adobe Photoshop), 3) if they are using a specific web browser with which SketchExpress has been developed and rigorously tested, and 4) if they have sufficient time to complete the entire study (which ranged from 30 to 60 minutes). If one or more of the answers were negative, they were paid only for filling out the pre-screening survey (a flat rate of $0.30). If they were eligible for the study, they were directed to a tutorial video made for the specific condition they were in (max 4 minutes).

Once they finished watching the video, workers were routed from a retainer pool to the task interface in advance to ensure they are available when needed. Once at the task page, workers were on standby for the span of multiple requests (from **IB1** to **IB9**) for a single session. Workers were paid a base rate of $10.20 per hour. At the beginning of each session, all participants were given a brief introduction to the experiment and were asked to familiarize themselves with the application by exploring what was covered in the tutorial video until they felt comfortable using the tool (warm-up time). During the warm-up time, workers were not given specific instructions unless they asked for clarification. At the end of the warm-up time, the requester checked if workers knew how to use the set of functions that were required to solve the tasks, which was included in our measure of warm-up time duration.

Experimental Design

Our study had three experimental conditions: **(C1)** the control condition, which used manual demonstration only (recreating [17]), **(C2)** the demo-and-replay condition where the application let workers record and replay animations but had no remix function, **(C3)** and the demo-remix-replay condition (the SketchExpress condition) that contained all proposed system features. Comparing the control condition ((**C1**)) with the other two ((**C2**)/(**C3**)) allowed us to understand the effectiveness of the demonstrate-remix-and-replay approach.We include an intermediate condition (**C2**) in order to account for the potential improvement (or detriment) in completion time or the accuracy. Our control condition ((**C1**)) reflects the original model used in the previous work, in which crowd workers listen and respond to demonstrate the described behaviors [17].

Each crowd worker was randomly assigned to one of the three experimental conditions (a between-subjects design) and each was asked to complete five tasks. Though the order in which the five tasks were presented was randomized, the order of the interactive behaviors within each task was fixed in the order presented above. Three workers left their session without completing all the tasks. This led us to recruit more workers so each condition was completed by the same number of workers. For each condition we had output data for the five tasks and nine animations we described earlier in the paper.

More specifically, tasks were conducted in the following order:

First Demo. A requester described an interactive behavior verbally and asked crowd workers to demonstrate (C1) or create an animation for it (C2, C3). Once the crowd believed they were done with the demonstration, they gave a "done" signal to the requester by changing the color of a circle (from red to green). If a worker asked any clarification question, the requester repeated the description one more time.

Second Demo. Once all interactive behaviors in a task were completed, the requester asked workers to demonstrate each behavior once more. Workers could use recorded (and remixed) animations in C2 and C3, while in C1 workers had to manually demonstrate the behavior each time. We asked workers to demonstrate behaviors twice in order to validate the benefits of reduced time for replayable interactive behaviors with a button click compared to manual demonstration.

Performance Measures

For each interactive behavior, we measured the accuracy by calculating precision and recall. Precision and recall indicate the overall quality of created animations according to the verbal description given to the workers. This was done based on scoring rubrics that we created to evaluate the crowd workers' demonstrations. The rubrics were created based on the verbal description that we provided to the crowd workers[3]. The annotators countered potential bias using well-defined yes/no questions, not subjective ones. Some of the examples include:

"Do the operations happen in the correct order?" (Task 1,2,4,5),

"Does Super Mario move forward?" (Task 1),

"Is there only one light (at least, and at most) on at any point in time?" (Task 2).

To calculate precision and recall, annotators counted the number of rubrics that were satisfied (True Positive), the number of rubric entries that workers missed (False Negative), and the number of unnecessary actions in the animation (False Positive). In addition, we annotated animations' start and end times, as well as other relevant events, such as replay, record, remix, requests for clarification, and reset (if available in the condition). These timestamps were used to calculate the average time spent completing each request. We manually annotated both the time and accuracy for three videos and assessed the consistency of these quantitative measurements by calculating intraclass correlations (ICC). The annotators had perfect agreement on precision and recall (ICC: 1.00) and nearly perfect agreement on time annotation (ICC: 0.99). We also annotated how long each participant spent getting familiar with the tool (warm-up).

RESULT: A SKETCH THAT BEHAVES

Now we discuss SketchExpress' quality and latency performance in the context of our two requests (First Demo and Second Demo). To examine the statistical significance, we ran a pairwise 2-tail *t-test* between two conditions (resulting in a total of three t-tests per metric).

Improving Quality

Figure 5 shows that SketchExpress significantly improves animation quality, resulting in 90.0% ($\sigma = 11.2\%$) overall recall for our system condition (C3) – a 27.3% improvement compared to the control condition (C1, 62.7%, $\sigma = 19.8\%$ $p < 0.001$). However, the recall in the demo-replay function without remix (C2) was not significantly different from the control condition (C1). This indicates that when the requested behaviors are complex, the addition of a remix function is critical for improving recall. On the other hand, when creating the simplest behavior, (**IB1**, which needed only one operation), there was not a significant difference between C1 and C3. Observationally, this is because the simpler behavior could be manually demonstrated accurately, thus there was not much room for improvement using remix.

There were significant improvements in precision across all conditions. Having replay function leads to a gain of

[3]Our rubrics are available for download here: https://sketchexpress.github.io/rubrics.html

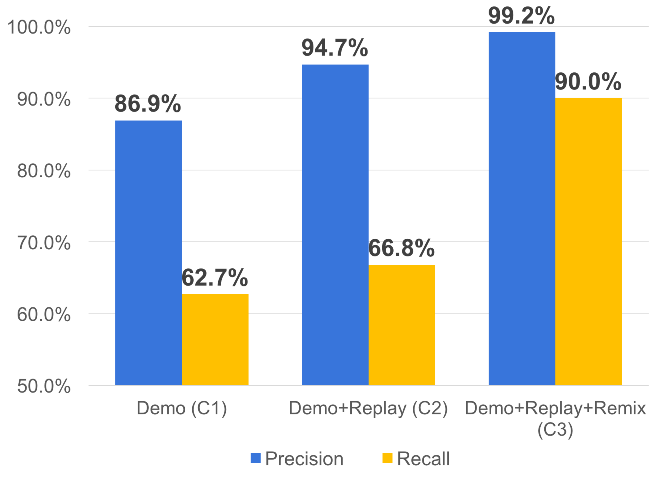


Figure 5. Though there is not a significant difference in recall between (C1) and (C2), recall is significantly higher in (C3) compared to (C1) and (C2) (both *p*<.001). There are significant increases in precision from C1 to C2 (*p*<.001) and from C2 to C3 (*p*<.001).

7.8% in precision, and the effect was statistically significant ($p < 0.001$). Precision in the system condition (C3) is almost perfect (99.2%, $\sigma = 2.9\%$), which is not surprising given that workers could choose to *skip* unnecessary operations in the remix mode, which was not available in (C1) and (C2). Remixing results in a 12.3% increase in precision when comparing (C3) to the control condition (C1,$\sigma = 20.7\%$, $p < 0.001$).

When replay functionality was available, the adoption rate was very high: when a requester asked for the behavior to be demonstrated the second time, we observed that *all* participants with access to replay (C2, C3) chose to use it for the animation they already created, instead of performing a new demonstration. The resulting sketch in the control condition (C1) was a static drawing of a graphical user interface that does not contain the interactive behaviors during the session, the sketches in (C2) and (C3) included behaviors that can reproduce the behaviors that were described in future sessions.

Improving Long-Term Latency

The First Demo result shows that, as anticipated, it took significantly more time to demonstrate *and* remix a demonstration (C3, 174.5s, $\sigma = 114.4s$) on average than it did to just perform the demonstration itself (C1, 39.4s, $\sigma = 46.7s$, $p < 0.001$). Even just recording the demonstration and replaying it to review added time compared to the control condition (C2, 78.3s, $\sigma = 60.6s$, $p < 0.001$). In addition, workers spent more time warming-up to the demo-remix-replay condition (C3, 10.2 min, $\sigma = 1.99m$) on average than they did in the control condition (C1, 5.44 mins, $\sigma = 3.49m$, $p < 0.05$). There was not a significant difference in warm-up time between the remaining two conditions (C2, 6.9 mins, $\sigma = 3.67m$).

While the initial creation of higher quality animations takes more time, once the behavior is recorded and remixed, it allows workers to respond very quickly to requests by replaying existing behaviors. The average time it takes to perform an animation the second time is 35.6s, 19.1s, and 12.4s in (C1), (C2), and (C3), respectively. Importantly, the demonstration speed in (C3) was significantly faster than it is in the control condition (C1, $p < 0.05$). The main source of this difference comes from the methods used to restore the initial state, which is depicted by the green portion of the graph in Fig.6. In the control condition (C1), a participant needed to manually restore the state to re-demonstrate a behavior on the canvas, while in the other two conditions, participants reset the canvas using the 'Reset' button. This is especially effective in the demo-remix-replay condition (C3) as workers frequently reset the state while they are remixing an animation. This result has implications for interface prototypes that utilize the same animation multiple times. Having both remix and replay functions potentially makes the creation of prototypes that use the same complex behavior multiple times even more efficient.

Task Engagement

One interesting observation we made was that crowd workers constantly tried to refine the animation in two conditions with demo-remix-replay (C2, C3), indicating high task engagement. The number of trials, the number of requests to clarify the user request, and the number of replays of the intermediate results is higher in the demo-remix-replay condition than in the other two (if available). We even witnessed several instances of crowd workers "rehearsing" the demonstration before recording. Some crowd workers spent additional time re-demonstrating and remixing the animation in (C3) even after generating a sufficiently accurate animation. In these cases, we are not exactly sure why the workers kept trying to re-demonstrate and refine the behavior as the animation generated was already good-enough, but some workers spent an excessively long time on the task (e.g., maximum time: 13.3 minutes for one animation), though previously work would categorize this type of worker as an "eager beaver" [4]. This indicates that we need a way to interrupt the excessive improvement to avoid wasting effort, but the natural desire to improve on the reusable components is promising.

In general, workers appear to be actively engaged with the tasks. Though we did not have a formal survey, five crowd workers voluntarily provided positive feedback about the task in the chatbox. To name a few: *"This is fun"*, *"It was a great study"*, *"Wow, this is great. I become an animator."*, *"How can I download animation on my PC?"* This is promising because if crowd workers find these kinds of tasks more engaging than other tasks available on MTurk, it will be easier to re-recruit participants who have already used our interface, allowing workers to gain expertise in the task over time.

CONCLUSIONS AND FUTURE WORK

This paper presents the design, implementation, and evaluation of SketchExpress, a system that enables non-expert crowd workers to quickly and easily create replayable animations that persist in electronic sketches. This, in turn, helps requesters more effectively prototype interactive UI behaviors. Crowd "Wizards-of-Oz" can quickly and accurately create replayable animations in minutes with a 27% improvement in recall.

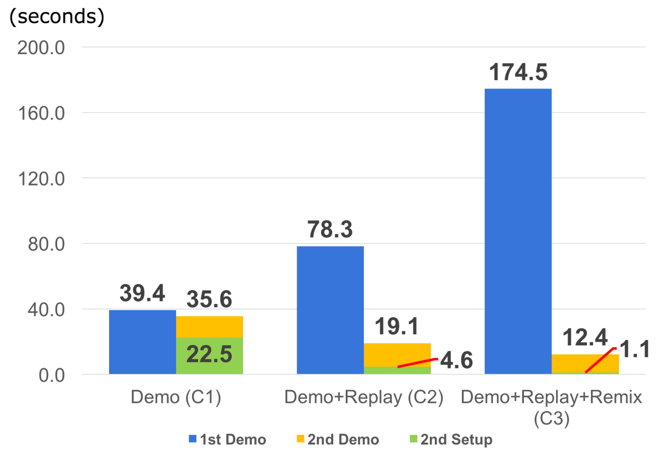


Figure 6. Latency of the 1st demo(blue) and the 2nd demo(yellow); The time it takes to demo-remix-replay an animation is longer than the other two conditions, but once an animation is created (the 2nd demo), a worker can respond quickly by replaying the animation. This is because the amount of time needed to respond to the demonstration request is the time it takes to restore the initial state needed to reproduce the requester behavior (the portion of the green bar in the yellow one).

Future work aims to better understand requester's side of the interaction. Our study controlled for requester variation by carefully generating and reading a script that reflects the target use cases. However, in real-world settings, requesters' verbal descriptions can vary depending on a number of factors (e.g., expertise, trust in crowdsourcing) and crowd workers may interpret them incorrectly. This poses a broader question of how requesters communicate with groups of non-expert crowd workers. We hope to potentially identify and learn from the verbal and visual cues requesters utilize to communicate with crowd workers. This can help system builders better understand the characteristics that make SketchExpress more or less useful, helping them to generalizing our approach to other nearly real-time crowdsourcing systems [5, 6, 31].

In addition, future work may explore how the system can learn from different instances of behavior created by multiple crowd workers, as well as edits made in the refinement steps to generalize a class of animation into an interactive behavior. Ideally, the system can automatically vary the animated behavior by itself depending on the different system states and settings. For example, Super Mario may jump differently if there is a brick wall in front of him compared to if the path ahead is clear. In the future, we plan to use machine learning to learn the structure of interactive behaviors and analyze the crowd's demonstrations based on requesters' verbal descriptions of behaviors. Eventually, we hope to develop a computational system that can help automate the creation process through the use of both human and machine intelligence.

ACKNOWLEDGEMENTS

This project was supported by the University of Michigan MCubed program. We thank Mark W. Newman, and Brandon Keelean for their input, as well as all of our study participants.

REFERENCES

1. 2017. Amazon's Mechanical Turk. (2017). http://www.mturk.com.
2. Connelly Barnes, David E. Jacobs, Jason Sanders, Dan B Goldman, Szymon Rusinkiewicz, Adam Finkelstein, and Maneesh Agrawala. 2008. Video Puppetry: A Performative Interface for Cutout Animation. In *ACM SIGGRAPH Asia 2008 Papers (SIGGRAPH Asia '08)*. ACM, New York, NY, USA, Article 124, 9 pages. DOI: http://dx.doi.org/10.1145/1457515.1409077
3. Michael S. Bernstein, Joel R. Brandt, Robert C. Miller, and David R. Karger. 2011. Crowds in Two Seconds: Enabling Realtime Crowd-Powered Interfaces. In *User Interface Software and Technology (UIST)*. 33–42. DOI: http://dx.doi.org/10.1145/1866029.1866080
4. Michael S. Bernstein, Greg Little, Robert C. Miller, Björn Hartmann, Mark S. Ackerman, David R. Karger, David Crowell, and Katrina Panovich. 2010. Soylent: a word processor with a crowd inside. In *User Interface Software and Technology (UIST)*. 313–322. DOI: http://dx.doi.org/10.1145/1866029.1866078
5. Jeffrey P. Bigham, Chandrika Jayant, Hanjie Ji, Greg Little, Andrew Miller, Robert C. Miller, Robin Miller, Aubrey Tatarowicz, Brandyn White, Samual White, and Tom Yeh. 2010. VizWiz: nearly real-time answers to visual questions. In *User Interface Software and Technology (UIST)*. 333–342. DOI: http://dx.doi.org/10.1145/1866029.1866080
6. Yan Chen, Sang Won Lee, Yin Xie, YiWei Yang, Walter S. Lasecki, and Steve Oney. 2017. Codeon: On-Demand Software Development Assistance. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17)*. ACM, New York, NY, USA, 6220–6231. DOI: http://dx.doi.org/10.1145/3025453.3025972
7. Allen Cypher, Daniel C. Halbert, David Kurlander, Henry Lieberman, David Maulsby, Brad A. Myers, and Alan Turransky (Eds.). 1993. *Watch What I Do: Programming by Demonstration*. MIT Press, Cambridge, MA, USA.
8. Richard C. Davis, Brien Colwell, and James A. Landay. 2008. 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, 413–422. DOI: http://dx.doi.org/10.1145/1357054.1357122
9. Richard C. Davis, T. Scott Saponas, Michael Shilman, and James A. Landay. 2007. SketchWizard: Wizard of Oz Prototyping of Pen-based User Interfaces. In *User Interface Software and Technology (UIST)*. 119–128. DOI:http://dx.doi.org/10.1145/1294211.1294233
10. Mitchell Gordon, Jeffrey P. Bigham, and Walter S. Lasecki. 2015. LegionTools: A Toolkit + UI for Recruiting and Routing Crowds to Synchronous Real-Time Tasks. In *Adjunct Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology (UIST '15 Adjunct)*. ACM, New York, NY, USA, 81–82. DOI: http://dx.doi.org/10.1145/2815585.2815729
11. Saul Greenberg, Sheelagh Carpendale, Nicolai Marquardt, and Bill Buxton. 2011. *Sketching user experiences: The workbook*. Elsevier.
12. Philip J. Guo. 2013. Online Python Tutor: Embeddable Web-based Program Visualization for Cs Education. In *Proceeding of the 44th ACM Technical Symposium on Computer Science Education (SIGCSE '13)*. ACM, New York, NY, USA, 579–584. DOI: http://dx.doi.org/10.1145/2445196.2445368
13. 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 (UIST '14)*. ACM, New York, NY, USA, 395–405. DOI: http://dx.doi.org/10.1145/2642918.2647375
14. Joy Kim, Justin Cheng, and Michael S. Bernstein. 2014. Ensemble: Exploring Complementary Strengths of Leaders and Crowds in Creative Collaboration. In *Proceedings of the 17th ACM Conference on Computer Supported Cooperative Work &#38; Social Computing (CSCW '14)*. ACM, New York, NY, USA, 745–755. DOI: http://dx.doi.org/10.1145/2531602.2531638
15. Aniket Kittur, Bongwon Suh, Bryan A. Pendleton, and Ed H. Chi. 2007. He Says, She Says: Conflict and Coordination in Wikipedia. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '07)*. ACM, New York, NY, USA, 453–462. DOI:http://dx.doi.org/10.1145/1240624.1240698
16. James A. Landay and Brad A. Myers. 1995. Interactive Sketching for the Early Stages of User Interface Design. In *Human Factors in Computing Systems (CHI)*. ACM Press/Addison-Wesley Publishing Co., 43–50. DOI: http://dx.doi.org/10.1145/223904.223910
17. Walter S. Lasecki, Juho Kim, Nick Rafter, Onkur Sen, Jeffrey P. Bigham, and Michael S. Bernstein. 2015. Apparition: Crowdsourced User Interfaces That Come to Life As You Sketch Them. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI '15)*. ACM, New York, NY, USA, 1925–1934. DOI: http://dx.doi.org/10.1145/2702123.2702565
18. Walter S. Lasecki, Kyle I. Murray, Samuel White, Robert C. Miller, and Jeffrey P. Bigham. 2011. Real-time Crowd Control of Existing Interfaces. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology (UIST '11)*. ACM, New York, NY, USA, 23–32. DOI: http://dx.doi.org/10.1145/2047196.2047200

19. Walter S. Lasecki, Phyo Thiha, Yu Zhong, Erin Brady, and Jeffrey P. Bigham. 2013. Answering Visual Questions with Conversational Crowd Assistants. In *Proceedings of the 15th International ACM SIGACCESS Conference on Computers and Accessibility (ASSETS '13)*. ACM, New York, NY, USA, Article 18, 8 pages. DOI:http://dx.doi.org/10.1145/2513383.2517033

20. Matthew Lease, Jessica Hullman, Jeffrey P. Bigham, Michael S. Bernstein, Juho Kim, Walter S. Lasecki, Saeideh Bakhshi, Tanushree Mitra, and Robert C Miller. 2013. Mechanical turk is not anonymous. (2013). http://dx.doi.org/10.2139/ssrn.2228728

21. S. W. Lee and J. Freeman. 2013. Real-Time Music Notation in Mixed Laptop Acoustic Ensembles. *Computer Music Journal* 37, 4 (Dec 2013), 24–36. DOI: http://dx.doi.org/10.1162/COMJ_a_00202

22. Sang Won Lee, Jason Freeman, Andrew Colella, Shannon Yao, and Akito Van Troyer. 2012. Evaluating Collaborative Laptop Improvisation with LOLC. In *Proceedings of the Symposium on Laptop Ensembles and Orchestras*. 55–62.

23. James Lin, Mark W. Newman, Jason I. Hong, and James A. Landay. 2000. DENIM: Finding a Tighter Fit Between Tools and Practice for Web Site Design. In *Human Factors in Computing Systems (CHI)*. 510–517. DOI:http://dx.doi.org/10.1145/332040.332486

24. Greg Little, Lydia B. Chilton, Max Goldman, and Robert C. Miller. 2010. TurKit: human computation algorithms on mechanical turk. In *User Interface Software and Technology (UIST)*. 57–66. DOI: http://dx.doi.org/10.1145/1866029.1866040

25. Mark MacKay. 2017. Method Draw. https://github.com/duopixel/Method-Draw. (2017).

26. Lennart Molin. 2004. Wizard-of-Oz Prototyping for Co-operative Interaction Design of Graphical User Interfaces. In *Proceedings of the Third Nordic Conference on Human-computer Interaction (NordiCHI '04)*. ACM, New York, NY, USA, 425–428. DOI: http://dx.doi.org/10.1145/1028014.1028086

27. B. Myers, S. Y. Park, Y. Nakano, G. Mueller, and A. Ko. 2008. How designers design and program interactive behaviors. In *2008 IEEE Symposium on Visual Languages and Human-Centric Computing*. 177–184. DOI:http://dx.doi.org/10.1109/VLHCC.2008.4639081

28. Brad A. Myers, Andrew J. Ko, and Margaret M. Burnett. 2006. Invited Research Overview: End-user Programming. In *CHI '06 Extended Abstracts on Human Factors in Computing Systems (CHI EA '06)*. ACM, New York, NY, USA, 75–80. DOI: http://dx.doi.org/10.1145/1125451.1125472

29. Brad A. Myers, Richard McDaniel, and David Wolber. 2000. Programming by Example: Intelligence in Demonstrational Interfaces. *Commun. ACM* 43, 3 (March 2000), 82–89. DOI: http://dx.doi.org/10.1145/330534.330545

30. Bonnie A Nardi. 1993. *A small matter of programming: perspectives on end user computing*. MIT press.

31. Michael Nebeling, Alexandra To, Anhong Guo, Adrian A. de Freitas, Jaime Teevan, Steven P. Dow, and Jeffrey P. Bigham. 2016. WearWrite: Crowd-Assisted Writing from Smartwatches. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*. ACM, New York, NY, USA, 3834–3846. DOI: http://dx.doi.org/10.1145/2858036.2858169

32. Željko Obrenovic and Jean-Bernard Martens. 2011. Sketching Interactive Systems with Sketchify. *ACM Trans. Comput.-Hum. Interact.* 18, 1, Article 4 (May 2011), 38 pages. DOI: http://dx.doi.org/10.1145/1959022.1959026

33. E. Sohn and Y. C. Choy. 2012. Sketch-n-Stretch: Sketching Animations Using Cutouts. *IEEE Computer Graphics and Applications* 32, 3 (May 2012), 59–69. DOI:http://dx.doi.org/10.1109/MCG.2010.106

34. Jaime Teevan, Shamsi T. Iqbal, and Curtis von Veh. 2016. Supporting Collaborative Writing with Microtasks. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*. ACM, New York, NY, USA, 2657–2668. DOI: http://dx.doi.org/10.1145/2858036.2858108

35. Bret Victor. 2012. Inventing on principle. (2012).