DemoWiz: Re-Performing Software Demonstrations for a Live Presentation

Pei-Yu (Peggy) Chi[1,2], Bongshin Lee[1], Steven M. Drucker[1]
[1]Microsoft Research
{bongshin, sdrucker}@microsoft.com
[2]University of California, Berkeley
peggychi@cs.berkeley.edu

ABSTRACT
Showing a live software demonstration during a talk can be engaging, but it is often not easy: presenters may struggle with (or worry about) unexpected software crashes and encounter issues such as mismatched screen resolutions or faulty network connectivity. Furthermore, it can be difficult to recall the steps to show while talking and operating the system all at the same time. An alternative is to present with pre-recorded screencast videos. It is, however, challenging to precisely match the narration to the video when using existing video players. We introduce DemoWiz, a video presentation system that provides an increased awareness of upcoming actions through glanceable visualizations. DemoWiz supports better control of timing by overlaying visual cues and enabling lightweight editing. A user study shows that our design significantly improves the presenters' perceived ease of narration and timing compared to a system without visualizations that was similar to a standard playback control. Furthermore, nine (out of ten) participants preferred DemoWiz over the standard playback control with the last expressing no preference.

Author Keywords
Demonstration; demo; software demo; presentation; video

ACM Classification Keywords
H.5.m. Information interfaces and presentation (e.g., HCI): Miscellaneous.

INTRODUCTION
Performing a software demonstration can be an effective way to communicate with the audience during a live presentation. By illustrating actions within a working system, presenters can guide the audience through an interaction flow and show results in real time. However, it is not always easy to perform an effective live demo. Problems such as software crashes, network connectivity issues, and configuration changes (e.g., screen resolution) may break a demonstration. Furthermore, talking while interacting with the system creates a high cognitive load on presenters. In addition, the stress of public speaking, especially during a high-stakes presentation, makes it difficult for presenters to deliver effective messages in a timely manner without forgetting to cover a set of core values of the system. An alternative is to present with pre-recorded screencast videos that capture the correct flow and information. Even though technical problems are less likely to occur with a video, it is challenging for presenters to talk over a video with appropriate timing because they have to mainly rely on their memories for the sequence and timing of interactions. Such a "canned" demo can often result in a less understandable or engaging presentation when a video is not tightly prepared to attract the audience's attention to anticipate the results [6].

The presenter view in PowerPoint or Keynote attempts to help presenters during slide show presentations by showing notes along with an upcoming slide. A teleprompter, commonly used for news programs or political speeches, prompts presenters with an electronic visual text of a speech or script. With this, speakers can appear to be speaking spontaneously as they look at the audience while reading the script. Inspired by these tools, we built DemoWiz (Figure 1), a system that assists presenters in giving software demonstrations with a screencast demo video during a live presentation. DemoWiz augments a screencast video with visualizations, enabling presenters to *anticipate* the video content rather than react to it; overlaying glyphs to guide presenters to the next action along with the time remaining before the action occurs.

DemoWiz supports the entire authoring process from capturing a screencast video; to rehearsing it and adjusting timings; to performing live presentation of the demo. During the recording phase, DemoWiz captures the screen pixels and logs input events, including event types and locations with timestamps. This event information is then processed and provided to presenters in the form of an adjustable timeline of events. During the rehearsal phase, presenters can speed up or slow down specific segments while navigating through the video recording using the timeline. In addition, they can add *pause markers* and short *text notes*. During the presentation, similar to current presentation tools like PowerPoint and Keynote, DemoWiz shows two views–one for the presenter and the other for the audience. The *Presenter View* is augmented with timed notes and a visualization of the captured events to help presenters synchronize their narration with the video.

CHI 2014, April 26 - May 01 2014, Toronto, ON, Canada

http://dx.doi.org/10.1145/2556288.2557254

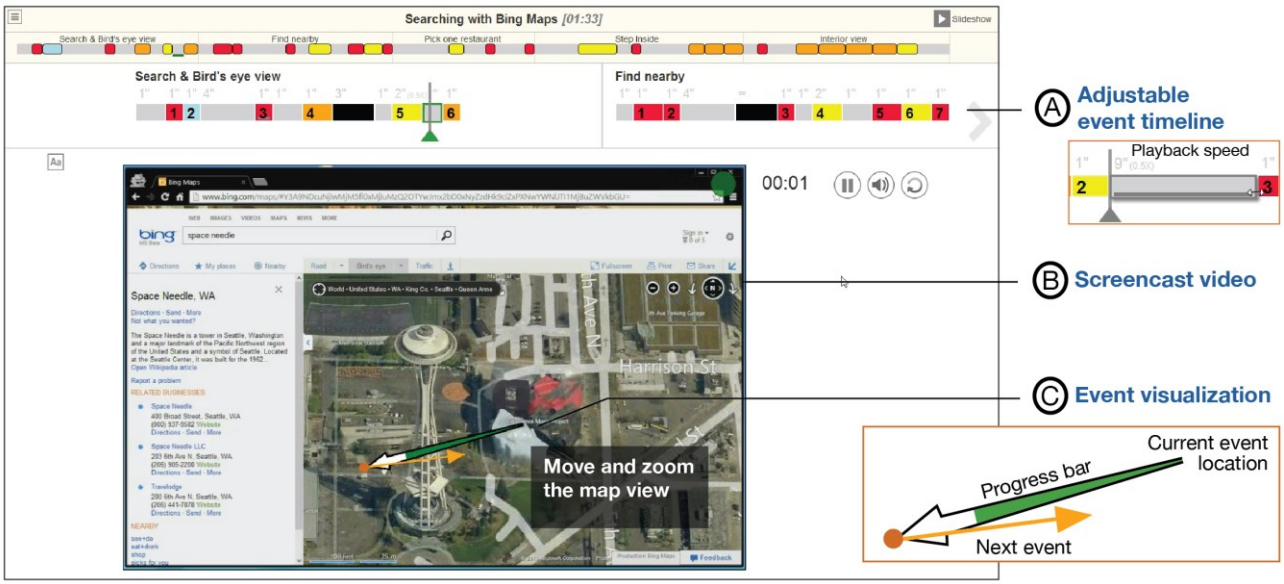

Figure 1. DemoWiz visualizes input events in a screencast video to help presenters anticipate the upcoming event for narrating a software demonstration in a live presentation.

To explore the effectiveness of the DemoWiz system, we performed a user study, comparing it with a version similar to a conventional video player. Our results show that, with DemoWiz, participants anticipated upcoming actions better and rated themselves as having narrated the video better. Moreover, 9 out of 10 participants preferred DemoWiz to a system without visualizations.

The contributions of this work are:

- An interactive video playback interface to help presenters control demo videos during a live presentation. It is combined with visual augmentation of screencast videos to enable presenters to anticipate upcoming actions and to be better aware of timing for narration.
- A lightweight workflow for presenters to record, rehearse and edit, and present demo videos. To support automatic video segmentation, we employ a hybrid approach to combine screencast videos and input event logs.
- Evaluation of the overall effectiveness of DemoWiz, incorporating visualizations into the presenter view of a video, across the workflow.

RELATED WORK

Workflow Capturing and Tutorials

There has been a considerable amount of research and many commercial tools devoted to revealing input events and operation sequences for software applications. Researchers have shown that visualizing input events in real-time during operations can provide better learnability of applications [8]. Tools such as Mouseposé[1] and ScreenFlow[2] capture mouse and keyboard events and apply special effects, such as drawing a circle around a mouse cursor. Workflows can also be captured from existing screencast videos [1] and screenshots [34]. In addition to visually enhancing events, presenting operation history helps users review the workflow. Approaches include annotating screenshot images with markers and arrows [15, 29], showing a list of before and after thumbnails and video clips [17], and creating a union graph of operations for workflow comparison [21]. These projects demonstrate the benefits of recognizing and visualizing events. Our work is related in that we use the stream of input events, but is focused on enhancing *a speaker's* experience in a live presentation by visualizing events *in advance* of the happening moments.

Another closely related area is the design of various tutorial formats that help viewers operate an interactive system. Work includes embedding video snippets in application tooltips [16], mixed-media tutorials that combine operation-based video segments with text descriptions and screenshots [5], and application-in-tutorial design enhanced by community-shared workflows [22]. These designs show possible ways for viewers to explore application features interactively, but again, differ from our goal of real-time assistance for presenters.

Visualizing and Navigating Video Content

Videos can be navigated at the content level beyond log events, such as visualizing subject movements in a storyboard design [12] and enabling direct manipulation of a target in 2D [9, 13, 20] or 3D [30]. These techniques help viewers understand content flow and playback videos, and have been applied to screencast videos [7]. It is also possible to automate video control based on user actions for scenarios such as operating software applications [31] and block assembling tasks [18]. Such novel forms of video navigation

[1] Mouseposé http://www.boinx.com/mousepose

[2] ScreenFlow http://www.telestream.net/screenflow

inspired us to explore new visual designs for revealing the video content that support live presentations.

Presentation Tools

Modern presentation tools have supported embedding video recordings and animation. Recent research has proposed advanced designs for content creation and navigation beyond simple slideshow composition, including: tools that help presenters compose content in a large canvas [14] as a path [26] or a directed graph [33] derived from zoomable graphical interfaces [3]; structure slides using markup languages [11] or sketching [25]; and define animation programmatically [35]. There has also been work on analyzing slide content for search and reuse [4, 32] and comparing revisions in a design process [10]. Our work shares similar goals of structuring a presentation based on event inputs that can be navigated and edited. However, we focus more on presentation enhancements of video content specifically for software demonstrations rather than on the authoring experience of the presentation itself.

Research on presenting information that can be perceived at a glance [28] helps presenters recall the content during a presentation, such as a callout to show finer resolution of an overall view [2]. Closely related, Time Aura provides ambient cues of a pile of slides using colors and a timeline for pacing [27]. Recent research shows that people like to have better control of the presentation even though it requires more effort [23], and earlier studies suggest that designing an integrated presentation tool for complicated tasks could be challenging [19]. These findings inspired our design on revealing content of a demo video with information that can be perceived with minimum attention.

DEMOWIZ DESIGN

To motivate and inform the design of a tool to support live presentations, we collected preferences for software demonstrations using an online survey. We describe the three design goals derived from the survey results.

Understanding Demo Preferences

To understand both presenters' and audiences' preferences for performing and viewing system demonstrations, we conducted an online survey in a software company and a university research lab. Our goal was to collect people's feedback on giving and seeing software demonstrations during live presentations. We received 73 responses from researchers, graduate students, software engineers, and designers. Their main research areas include human-computer interaction (64.4%), software engineering (21.9%), and machine learning (20.6%); 66.7% were male. Among all the respondents, 35.6% indicated that they were very experienced at giving software demos to the audience during a live presentation; 46.6% had demoed at least once; 13.7% had not demoed but attended talks that showed a software demonstration.

We asked respondents who had demo experience (N = 60) how they preferred to perform a demo. Their answers were: a live demo (25 out of 60), pre-recorded videos (15), a mixed format of a live demo and videos (12), static screenshots (4), and other (4). In Table 1, we list the top 2-3 reasons for their preferences. Giving a *live demo* can be more engaging with a working system and match the audience's interests, but presenters can encounter unexpected problems and forget to show important features within a given time constraint. On the other hand, presenting with *a demo video* avoids such problems by extracting the most important parts, and can allow visual highlighting (labelling or zooming), but can be less engaging. In addition, it is hard to narrate.

We were also interested in reactions as an audience member. For respondents who had seen software demos (N = 70), we asked how they preferred to see the demonstration performed. We found a slightly different preference: a live demo (36 out of 70), a mixed format of a live demo and videos (24), pre-recorded videos (7), and other (3). However, the reasons were well aligned with presenters' concerns. A *live demo* shows a working system and can be more engaging, but the audience might need to wait for system problems to be resolved or sometimes see presenters rambling. A *demo video* can show the most important parts, sometimes assisted by visual highlighting, but it can be hard to tell which parts of a demo are real, and can be less engaging to the audience.

Design Goals

From the survey results, we understand that giving a live demo is often more preferable than showing demo videos. However, we cannot, in general, address some of the main concerns with giving a live demo – that is, stability of the software system and variations in the presentation environment which can cause the demo to fail. Therefore, we

	Presenters		Audience	
	Advantages	**Disadvantages**	**Advantages**	**Disadvantages**
Live Demo	• More engaging (88.3%) • Show a working system (86.7%) • Easy to adjust a demo based on audience's interests (45%)	• May encounter unexpected system problems (86.4%) • May forget to show important features (33.9%) • Hard to control time (35.6%)	• Show that it's a working system (97.1%) • More engaging (72.9%)	• May need to wait for problems to be solved (78.6%) • Presenters may end up rambling (37.1%)
Demo Video	• Avoid system problems (95%) • Work with a partially working system or a mockup (51.7%) • Can edit to remove mistakes or add highlights (51.7%)	• Less engaging (57.6%) • Hard to match their narration to the video content (30.5%)	• Avoid problems (81.4%) • Show the most important parts with visual highlights (60%) • Work with a partially working system or a mockup (45.7%)	• Hard to tell which parts are real (62.9%) • Want to see how the actual system works (44.3%) • Less engaging (37.1%)

Table 1: Survey of software demonstration preferences from presenters' (N=60) and audience's (N=70) point of views.

instead aim to address some of the drawbacks with demo videos while preserving their advantages. More specifically, our goal is to make demo videos *more engaging* by assisting presenters in adjusting their narration to guide the audience through the material. In this section, we describe our design goals to support more effective demo video presentations.

G1. Show what's coming next, where and when it will occur.
To engage the audience with the demonstration, it is important for presenters to guide the audience's attention to the right place at the right time. To do so, presenters should be fully aware of upcoming actions – specifically *what* actions will happen, *where* they will occur on the screen, and *when* they will happen.

G2. Minimize required attention or interpretation.
While it is our desire to help presenters understand and anticipate impending events, we should not overburden a presenter who is already narrating a specific set of talking points. As a tradeoff between providing more information and minimizing cognitive load, any augmentation of the video needs to be offered in a glanceable fashion, i.e., information can be interpreted quickly and without the presenter's full attention.

G3. Support light-weight editing during rehearsal.
Different presentations may require more or less extensive explanations, and when first recording a demo video, it may not be possible to perform the demo at the same rate necessary for a live presentation (e.g., typing can be difficult or system response times may be variable). In addition, it should be easy to review, practice, and modify the pace for a particular presentation. For all these reasons, lightweight editing and rehearsal are necessary.

Using these principles as a guiding rubric for our design, we iterated on several versions of the DemoWiz system.

DEMOWIZ

Augmented Workflow

For presenters to narrate "live" over a video recording, we propose augmenting a typical workflow from capturing a screencast video; to rehearsing it and adjusting timings; and finally to live presentation of the demo video (Figure 2).

DemoWiz first captures a screencast video and input events during a software demonstration from a user-defined rectangular region. Once the recording is done, DemoWiz analyzes the low-level event stream and transforms it into higher-level events such as mouse clicks, double-clicks, and drags. DemoWiz then allows presenters to edit the timing and notes while practicing their presentations with the presenter view equipped with an adjustable event timeline (Figure 1a). Finally, presenters can give a live presentation using the same UI (i.e., presenter view) and show the audience view without visualization to the viewers.

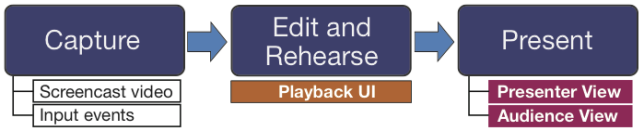

Figure 2. DemoWiz workflow: Presenters captures a software demonstration, edit the recording while rehearsing with our playback UI, and present the edited video to the audience using a presenter view.

Visualizations

To enable presenters to focus on their narration and the original video contents, DemoWiz augments the screencast recording by automatically overlaying simple glyphs.

Input Event Glyphs
DemoWiz overlays visual annotations of events on the screencast recording in a graphical way where the events happen. For example, in Figure 1, the presenter clicks and drags the map view to the right. DemoWiz uses the following simple, distinctive glyphs to differentiate event types as Figure 3 shows:

- Mouse click: a red circle with a radius of 20-pixels,
- Double-click: a green circle with a radius of 20-pixels,
- Mouse drag: a thin, orange line with a dot at the start point and an arrowhead at the end point,
- Mouse scroll: a thin, yellow line, 80 pixels long, with an arrowhead, and
- Keystrokes: text in blue.

At any given time during the video playback, DemoWiz shows the current event and the upcoming event on the video. We tried to show more than two events within a fixed time period in our initial prototypes. However, we noticed several issues. First, the view becomes too cluttered to understand at a glance, especially when the original video is visually complex. Second, it is not easy to convey the order of the events. Third, it is difficult to observe when multiple events are spatially close. Therefore, we provide minimum but essential events for recall.

Visual Guides to the Next Events
In order to help guide the presenter's attention, DemoWiz overlays a motion arrow between the current and upcoming events on the demo video (Figure 1c). This is inspired by storyboard design used in filming where an arrow effectively shows the movement of a camera or an actor in a single shot [12]. We expand the idea of guiding attention for a specific purpose: the arrow in DemoWiz shows the movement from one action (e.g., click a checkbox) to another action (e.g., click a button). By overlaying this motion arrow, the visualization matches the flow of a presenter's attention when they observe the video content.

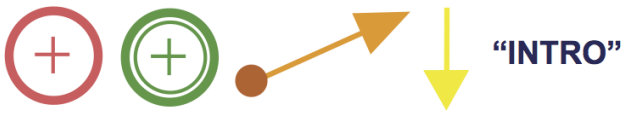

Figure 3. DemoWiz visualizes input events in a graphical way. From the left to right we show a mouse click, double-click, a drag, a mouse scroll, and keystroke events. These glyphs are overlaid on the video recordings.

Since the distance between two consecutive event segments vary, we created three visual designs to make sure the arrows are visible to lead a presenter's attention:

- For two events that are located far away (e.g., clicking an "OK" button after selecting a checkbox on a page), apply a *straight* arrow (Figure 4a).
- For events that are nearly at the same location (e.g., click the "Next" button twice to navigate a list of selections), apply a *round* arrow that points to the current location (Figure 4b).
- Otherwise, apply a *curved* arrow (Figure 4c).

Sense of Timing

DemoWiz provides a sense of timing for an upcoming action so that presenters can adjust their narration. First, DemoWiz embeds *a progress bar* in the motion arrow to show relative time (Figure 1c). The green bar shows the proportional time that has been passed before reaching the next event (Figure 5 top). When a motion arrow is filled up with green, it fades away and guides the presenter to the next action. We were concerned that people may associate the length of an arrow to the length of time. Therefore, we also incorporated a *countdown* visualization where circles will fade out in the last three seconds before the next action starts (Figure 5 bottom) to convey absolute timing.

Visualization Examples

Figure 6 presents examples of DemoWiz visualizations with four different systems. The glyphs effectively show the start and end points of mouse drags and the locations of mouse clicks. Motion arrows help direct the presenter's attention between events, such as start the end of the drag event to clicking a button (Figure 6a, 6b), clicking between several options (Figure 6c), or selecting a specific slide after scrolling down (Figure 6d).

Lightweight Editing During Rehearsal

During rehearsal for their demonstration, presenters can modify the video timing and add reminder notes for their narration. DemoWiz shows the type and length of each event in a sequence in a timeline (Figure 1a). Each segment is shown as a block whose width indicates its length in time. To simplify the timeline and avoid fine-grained adjustment, lengths of event blocks are rounded to the second. Presenters can modify the playback speed of a segment by dragging the boundaries of a segment on the timeline. For example, presenters can speed up to shorten long text inputs, and slow down for fast mouse drag inputs that select multiple objects.

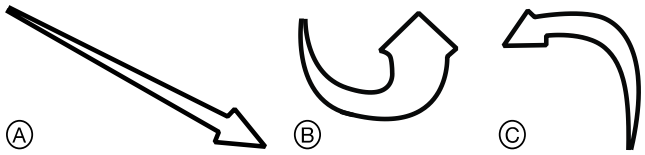

Figure 4. Three types of motion arrows in DemoWiz that guide presenters to the next event of different distances at a (A) far, (B) nearly the same, and (C) near location.

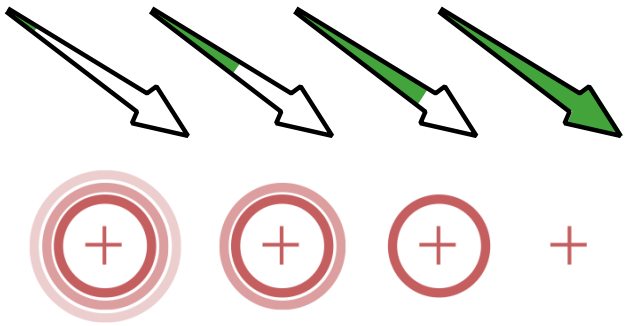

Figure 5. A progress in time guides the presenter from the current event (left) gradually to the upcoming action (right) using relative timing with a progress bar (top) and absolute timing (bottom).

Figure 6. Examples of DemoWiz visualizations with four different systems and input event sequences.

Sometimes a change in the playback speed may result in an awkward effect that is noticeable to the audience, especially when showing a UI transition. Therefore, DemoWiz supports two special time control markers to enable breaks in the narration. Presenters can add an adjustable *pause* segment, at which the system will pause at the last frame of the previous segment for the specified length of time. If presenters prefer full control on pause length, a *stop* marker ensures the video stays paused at the last frame of the previous segment and will not proceed until presenters manually resume the playback of the video.

DemoWiz enables presenters to add a short text note (such as the reminder "*Move and zoom…*" in Figure 1) so that they could remind themselves of upcoming actions at a higher level. The note can be positioned manually at any location on a video so that it does not block important video content, and will be shown for 3 seconds before the associated event.

For every edit that is associated with time changes (including playback speed and pauses), DemoWiz computes and updates the total presentation time as well as updating the progress bar and countdown to provide accurate timing.

Presenter View

During presentation, DemoWiz shows two views in separate windows. Presenters can observe visualizations using the presenter view, while the audience will see the audience view with a full-screen video that has no enhanced information. DemoWiz synchronizes the videos in both views based on

presenters' editing decisions to ensure the same playback speed and time. As with a conventional video player, presenters can control the video, to pause and play at any time. In addition, when a video is paused (or stopped), presenters can hover the mouse over the demo video in the presenter view to point out an important area, as many presenters currently do in a live demo. DemoWiz then simulates and synchronizes a mouse cursor in the audience view to help the audience follow the demonstration.

Implementation Details

During recording, DemoWiz captures the screen within a specified region and logs low-level system input data with *timestamps* (with an accuracy of 0.1 seconds) from the operating system, including:

- *Mouse events* (mouse downs, mouse ups, and mouse wheel movements) and their *positions* (in x-y coordinates relative to the screen-captured region).
- *Key-press events* (keyboard input).

Once presenters finish their demonstrations, DemoWiz analyzes the low-level event stream and transforms it into high-level event metadata. For mouse events, we pair each mouse down and up into mouse *clicks*, *double-clicks*, or *drags*. We group any consecutive mouse wheel events within a time threshold of 2 seconds to one *scroll* event and any key-press events within the same threshold to one *keystroke* event (e.g., combine keys d-o-w-n-t-o-w-n to "downtown"). For each high-level event, we log the *start* and *end time* (timestamps of the first and the last low-level event).

Based on the start and end times of these high-level events, DemoWiz segments the screencast video recording into *event segments*. Any gap between two consecutive input events is marked as an *inactive segment*, which may include mouse hovering, UI transitions of the demo system, or static frames with no visual changes. DemoWiz adjusts the boundaries of these event segments to avoid any short visual effect that cannot be observed. DemoWiz examines segments in a linear order to ensure each segment lasts at least t_{min} seconds long, which is set as one second based on our early testing. For an event segment S_i of time (t_{start}, t_{end}) that $t_{end} - t_{start} < t_{min}$, DemoWiz expands 0.5 second forward and backward if S_{i-1} and S_{i+1} are inactive. If the adjusted S'_{i-1} and S'_{i+1} are shorter than t_{min}, DemoWiz merges it to the shorter neighbor segment. Currently, DemoWiz does not analyze these inactive segments, but techniques including computer vision and video analysis [1, 5] can be applied for finer segmentation.

The capturing program is implemented in C#. Two APIs were used: 1) the Windows Event Log API for mouse and keyboard hooks and 2) the Expression Encoder 4 API for screen recording running on Microsoft Windows 7. The recorded metadata (stored in a JSON object) and screencast video (in MP4) are read by the Presenter UI, which is implemented using standard Web technologies, including HTML5, CSS3, JavaScript, and jQuery. In particular, the visualization is rendered on the canvas element on top of the video object on the fly based on the video playback time. The audience view is generated by the main browser window of presenter view for video control.

EVALUATION

To evaluate the DemoWiz design, we conducted a controlled experiment in which participants recorded and edited a demo video, and gave a presentation with the edited video. Specifically, we wanted to see if presenters would evaluate their own performances higher with the support of our augmented visualizations and control of timing.

Baseline Condition: DemoWiz without Visualization

Since DemoWiz allows for rapid editing of the video, it would have been unfair to compare it with a conventional video player without supporting any editing during the rehearsal phase. We therefore modified our system to serve as the baseline condition, providing participants with the same lightweight editing of the video in each condition. However, during presentation, the baseline condition was similar to a conventional video player that shows only the video without event timeline and augmented visualizations. It also did not support the *stop* markers and *text notes*, i.e., participants could only adjust playback speed of each segment and add variable length *pauses*. During presentation, participants only saw the video with a traditional timeline. They could, however, pause (or stop) and resume the video manually at any time during playback.

Study Design

We conducted the study as a within-subjects design in a usability room. After recording and editing a video using the same system, each presenter gave a presentation with both systems to an experimenter. To control the effect of order and learning, we prepared two tasks that included similar interaction flows and counterbalanced the order of the two systems—DemoWiz and Baseline—but we fixed the order of tasks. Even though presenting to a single audience member in a usability room is not the same as using the system with a large conference audience, it is important to control the tasks and presentation as closely as possible to understand the relative benefits of the system in comparison with a baseline condition.

For each condition, we observed and coded the *timing* of narration that matched the video content and noted the time in seconds when an event was described *before*, *at*, or *after* the action happened in the demo video. We also marked obvious *breaks* between narrations, *errors* when the narration was not about the current or following events (e.g., discussing actions in a different order than they actually occurred), and *misses* when an important action was not mentioned. To avoid unconscious bias that might influence the coding of the videos, we neutrally named the recordings and coded them all in a batch. We focused on objective timing measurements as much as possible, measuring deviation from specific video events and their corresponding

narrations down to a second. Finally, we gathered qualitative feedback through satisfaction and preference questionnaires.

Participants

We recruited 12 participants (10 males and 2 females) from a software company. However, we excluded the data from two participants (1 male and 1 female); one was due to a software bug during one condition and another was because the participant requested to restart a presentation in one condition. The average age of the effective 10 participants was 37.3 ranging from 24 to 64 years of age. We recruited participants who had experience at showing a software demonstration to an audience such as giving a presentation at a conference. Four participants were native English speakers and the rest were fluent in English. The expertise of participants included audio processing, computer graphics, human-computer interactions, machine learning, networking, and software engineering. Each participant was compensated with lunch coupons worth $20.

Procedure and Tasks

Each session consisted of one training task and two experimental tasks. For the training task, to introduce the common features for recording and editing the video, we designed a simple workflow of five steps to demonstrate editing of a slide using PowerPoint. The experimenter briefly demonstrated an example and then introduced the recording program that captured the screen. Participants were then asked to practice and record using the recording program.

The two tasks consisted of a similar sequence and interactions: 1) searching with Bing Maps to show the 2D map view and the Bird's Eye view, looking for a restaurant, and navigating to the interior view of a specific restaurant; and 2) searching with Google Shopping to show the search results with the Grid view, filtering and voting for reviews, and navigating the 3D product view of an espresso machine. For each task, we provided a specific scenario along with a list of subtasks. The experimenter walked through this list with participants to ensure that they could easily find the features that needed to be demonstrated. Participants were then asked to practice (3-5 minutes), record (about 2 minutes), and rehearse and edit (5-10 minutes).

To help simulate a conference setting where participants would not be able to present immediately after having recorded a demonstration, we inserted an intentional 1-minute gap between rehearsal and presentation. During this gap before giving the presentation, we asked participants to watch a conference showcase video. Participants were then asked to stand up and gave a 2-3 minute presentation to the experimenter in a usability room.

After each task, participants filled out a questionnaire of 8-10 questions asking about their experience (8 for the Baseline condition, and 10 for the DemoWiz condition). At the end of the session, an online questionnaire was provided for them to present overall preferences and leave comments. Each session lasted about 1.5 hours.

Experiment Setup

Each participant used a desktop computer running Windows 7, Expression Encoder 4 for screen recording, and a web browser for the DemoWiz user interface. A regular mouse and keyboard were provided, along with two 27-inch displays, one for editing (during rehearsal) and showing the audience view (during presentation), and the other for the presenter view on a stand-up table. The resolution of both displays was 1920×1200 pixels. The average captured screen area was 1311×857 pixels. In the presenter view, the video resolution was within 1000×600 pixels; in the audience view, the screencast videos were resized to fill the entire display with at least 100-pixel wide border in black. During the study, the experimenter stayed in the room, providing instructions and sitting behind the participants during the recording and editing phases.

Results

Ten participants successfully recorded, rehearsed, and gave a demo with both systems.

Subjective Preference

Figure 7 shows the average subject responses (on the 7-point Likert scale) from presenters for both systems. We analyzed these subjective responses using a Wilcoxon signed-rank test. We found significant differences in responses for ease of narration (DemoWiz $\mu = 6.2$ over Baseline $\mu = 4.5$, $p = .018$) and ease of presentation (6.4 over 5.2, $p = .048$). We also found marginally significant differences in participants' overall satisfaction with their presentations (5.5 over 4.7, $p = .062$). Participants also tend to agree that DemoWiz helped them interpret timing (6.1 over 4.4, $p = .067$).

In addition, 9 out of the 10 participants preferred DemoWiz to the system without visualization and would choose to present with DemoWiz if they were asked to give a public software demo; the remaining participant indicated no preference for both questions. The general feedback was also encouraging. For example, P1 commented "*Awesome system. I'd use it today.*" and P5 "*felt more confident in being able to present what I wanted to.*"

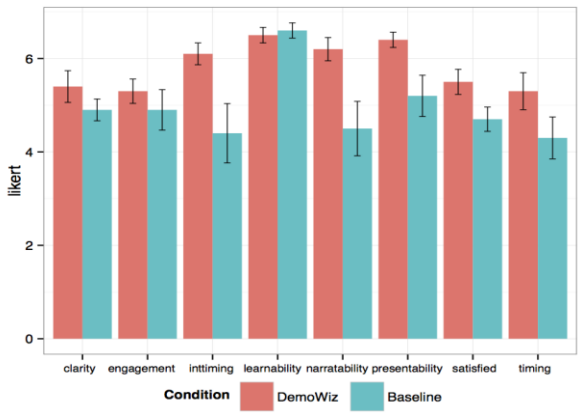

Figure 7. User feedback from questionnaire on the 7-point Likert scale.

Visualization as a Supportive Cue

Participants answered that they were able to understand DemoWiz visualization of input events (μ = 6.0) and found it supportive for their presentations (μ = 6.3). They also commented that the DemoWiz visualization supported the presentation in various aspects: "*the visualization reminds of the order of the content*" (P1), "*Really liked the ability to know what was coming up*" (P2), "*It provides better insight of the progress of the video*" (P6), and "*viz gave me an idea about timing or something I was going to forget to say*" (P9).

Narration Timing

We coded the 20 recordings of participants' final presentations to observe the timing of narration of each action in correspondence with the video content (11 key events for both tasks). With DemoWiz, participants tended to *anticipate* the upcoming events rather than talk afterwards, where the average timing was -0.1 seconds with DemoWiz (i.e., narrated the action before it happened) and 0.4 seconds with the Baseline condition (i.e., explained the action after it was shown). We found a significant difference in the number of times that events were anticipated by the narration, co-occurred, or occurred after the fact ($\chi^2(2,220) = 8.6$, $p = .01$, see Figure 8).

In general, this supports our suspicion that DemoWiz would help in anticipating an event as opposed to talking about it after it occurred. More important though, was how often a narrator spoke about an event within several seconds of when the event actually occurred. By defining *better* timing as when a presenter's explanation came within 2 seconds of a shown event (either prior, exact, or after), there was marginal significance by condition (p = .089 with DemoWiz performing better). In addition, with the Baseline condition, the timing of narration was less consistent and off more, varying from 6 seconds early or 10 seconds late with a variance of 3.9 seconds, in comparison to the DemoWiz condition with at most 3 seconds early to 3 seconds late and a variance of 1.9 seconds.

Five participants had an obvious *error* (forgot the next action or incorrectly narrated another action), had a long *break* (waiting for more than 2 seconds until the action was made), or *missed* an action (did not explain an important feature) when presenting with the Baseline condition. On the other hand, in the DemoWiz condition no errors were made, and there were only one long break and one miss from two different participants, respectively.

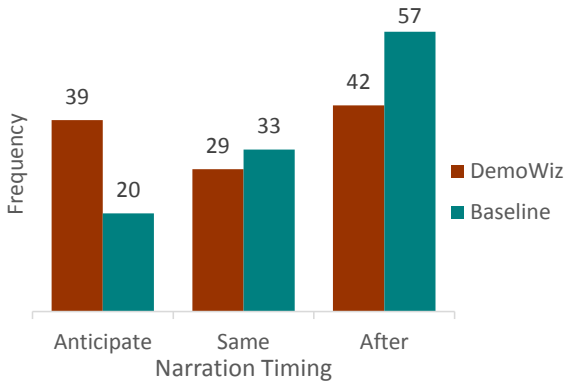

Figure 8. The number of times events were anticipated by the narration, co-occurred, or occurred after the fact.

Participants' comments also support the fact that DemoWiz helped presenters anticipate the upcoming events. P7 explained, "(I) *felt better able to time my speech to coincide with visual events, rather than trailing after them. Without the event visualizations, I felt like I was talking about what the audience had just seen, rather than having my words and visuals combine to a single message.*"

Editing Experience

We collected comments on the workflow. Participants found it easy to record (μ = 6.4) their demonstrations with DemoWiz. For editing features, they found it easy to edit in general (6.6), including controlling the playback speed (6.5) and adding pauses and stops (6.5), but it was less easy to add text notes (4.8); only two participants used this as reminders.

Although using different strategies, all of the participants adjusted the playback speed for matching their narration. Some sped up whenever possible and added stop markers for transitions; some slowed down the repetitive actions (such as drags) to demonstrate effects. P6 said, "*I really liked being able to add 'stop' events so I could 'fake' my demo better.*" DemoWiz made it easy for participants to separate the capturing and presentation preparation as P5 explained, "*Overall, recording was very easy. In fact, as I got to the second task, I realized that I really don't need to think about the words as I record because later on I will be able to slow down and speed up time …*"

On average, the length of demo videos was 2'09" before editing and 2'05" after editing, and the presentation was 2'38" long. Each participant spent 7.5 minutes on average to edit. For each demo of 44 segments on average, participants adjusted 3.15 segments for speedup and 4.25 segments for slowdown, and added 0.55 pause markers. In the DemoWiz condition, 1.2 stop markers and 0.2 text notes were added.

DISCUSSION AND FUTURE WORK

New Tool to Present Video in Real-Time

DemoWiz is an attempt to make demo videos more engaging by helping presenters anticipate the upcoming events rather than reacting to them, leveraging a refined workflow with augmented visualizations. Overall, participants liked the DemoWiz visualization, finding it supportive rather than distracting. For examples, P4 said, "*Event visualization was very powerful – definitely the way to go*." and P2 (who first experienced Baseline) was originally skeptical when he first saw the visualization but immediately found it helpful and not distracting. This corresponds with our goal of designing the visualization with a minimal cognitive load.

Editing Capabilities

Lightweight editing during rehearsal not only makes it easy to edit the recorded video but also lowers the burden of the

initial recording. Presenters do not have to prepare a complete script for exact timing. They also do not have to repeat recording many times to grab the best recording.

Some participants appreciated our design choice of providing only minimum but essential editing capabilities to make the process as light as possible. P2 mentioned that "*Ironically, I think it's better to have limited editing feature set -- this system was very easy to learn/use*." A few participants expressed the need for more editing features: P1 explained, "*(I wish the system could be) cutting events in parts so that I can slow down/speed up/remove portions of, e.g., a mouse trajectory*"; P3 wanted to "*flip segments around*" and P8 thought "*break up or merge blocks*" would be helpful. We found these interesting as the system enabled more possibilities, but there is a tradeoff between providing a powerful tool and lowering the burden in editing. We believe that this is a design choice that needs to be balanced.

Our system does not support combining two or more video clips for a presentation. Sometimes, presenters may also want to update part of the existing material to show new features of their developing systems. For example, P4 explained that he would like to see "*the ability to record multiple clips and insert them in a timeline."* This would be straightforward future work because the current DemoWiz framework is designed to be able to implement this.

Editing can still be limited to support fine timing control of narration. P10 explained, "*The length of narration changes each time I present, and it is difficult to perfectly align the timing.*" Automatically navigating a video based on presenters' performances could be an interesting avenue of exploration, similar to scenarios of following a tutorial [31] or performing music [24]. However, we decided not to pursue this approach because it would present its own form of risk relying on unreliable speech recognition during a live presentation. Also, considering the time constraints presenters usually have, we chose to provide full control for presenters rather than trying to intelligently update a video.

Study Audience Engagement

In our user study, we gathered presenters' opinions as to how engaging their presentation was, and we explored the relative timing of the narration to events in the video. Ultimately, however, our goal is to help increase audience engagement. Measuring audience engagement is an ongoing topic of research, and we would like to explore ways of quantifying the relative impact of the DemoWiz system, but that work was out of scope for this project.

Enhance the Audience View

Some participants commented that it would be helpful to highlight certain input events for the viewers to observe subtle changes. For example, P10 wanted to enable, "*visualize mouse events such as clicks and scrolls for the audience so they know what is going on."* The current DemoWiz framework makes it easy to achieve this goal by highlighting the audience view *only* when the event happens. In other words, presenters and audience will see different visual effects, where the former observe events in advance and the latter see a visualization synchronized with the demo video content.

Beyond Software Demonstrations

Although our current implementation is focused on software demonstrations, we argue that it is possible to expand our system design to more advanced inputs. By defining event types that a system recognizes (e.g., a pinch gesture on a multitouch device or a specific pose detected by a 3D sensor), it is possible to log the events and align them with the captured video for later use. In addition, the enhanced presentation mode can be potentially applied to other domains where knowing the timing and the sequence of events is crucial, such as narrating over animated presentation slides with dynamic graphical objects. DemoWiz is an important first step towards validating this general approach and we believe our work could inspire future research in these directions.

CONCLUSION

This paper introduces DemoWiz, a system with a refined workflow that helps presenters capture software demonstrations, edit and rehearse them, and re-perform them for an engaging live presentation. DemoWiz visualizes input events and guides presenters to see what's coming up by overlaying visual annotations of events on the screencast recording where the events happen in a screencast video. It also provides lightweight editing for presenters to adjust video playback speed, pause frames, and add text notes. A user study showed that DemoWiz was effective in helping presenters capture timing and narrate over a demo video.

REFERENCES

1. Banovic, N., Grossman, T., Matejka, J., and Fitzmaurice, G. Waken: reverse engineering usage information and interface structure from software videos. *UIST '12*, ACM Press (2012), 83–92.
2. Baudisch, P., Good, N., and Bellotti, V. Keeping things in context: a comparative evaluation of focus plus context screens, overviews, and zooming. *CHI '02*, ACM Press (2002), 259–266.
3. Bederson, B.B. and Hollan, J.D. Pad++: a zooming graphical interface for exploring alternate interface physics. *UIST '94*, ACM Press (1994), 17–26.
4. Bergman, L., Lu, J., Konuru, R., MacNaught, J., and Yeh, D. Outline wizard: presentation composition and search. *IUI '10*, ACM Press (2010), 209–218.
5. Chi, P.-Y., Ahn, S., Ren, A., Dontcheva, M., Li, W., and Hartmann, B. MixT: automatic generation of step-by-step mixed media tutorials. *UIST '12*, ACM Press (2012), 93–102.
6. Cohen, P. Great Demo! How to Create and Execute Stunning Software Demonstrations. iUniverse, Bloomington, IN, USA, 2005.

7. Denoue, L., Carter, S., Cooper, M., and Adcock, J. Real-time direct manipulation of screen-based videos. *IUI '13 Companion* (2013).
8. Dixon, M. and Fogarty, J. Prefab: implementing advanced behaviors using pixel-based reverse engineering of interface structure. *CHI '10*, ACM Press (2010), 1525–1534.
9. Dragicevic, P., Ramos, G., Bibliowitcz, J., Nowrouzezahrai, D., Balakrishnan, R., and Singh, K. Video browsing by direct manipulation. *CHI '08*, ACM Press (2008), 237–246.
10. Drucker, S.M., Petschnigg, G., and Agrawala, M. Comparing and managing multiple versions of slide presentations. *UIST '06*, ACM Press (2006), 47–56.
11. Edge, D., Savage, J., and Yatani, K. HyperSlides: dynamic presentation prototyping. *CHI '13*, ACM Press (2013), 671–680.
12. Goldman, D.B., Curless, B., Salesin, D., and Seitz, S.M. Schematic storyboarding for video visualization and editing. *ACM Trans. Graphics (SIGGRAPH '06) 25*, 13 (2006), 862–871.
13. Goldman, D., Gonterman, C., and Curless, B. Video object annotation, navigation, and composition. UIST *'08*, ACM Press (2008), 3–12.
14. Good, L. and Bederson, B.B. Zoomable User Interfaces as a Medium for Slide Show Presentations. *Information Visualization* 1, 1 (2002), 35–49.
15. Grabler, F., Agrawala, M., Li, W., Dontcheva, M., and Igarashi, T. Generating photo manipulation tutorials by demonstration. *ACM Trans. Graphics (SIGGRAPH '09) 28*, 3 (2009).
16. Grossman, T. and Fitzmaurice, G. ToolClips: an investigation of contextual video assistance for functionality understanding. *CHI '10*, ACM Press (2010), 1515–1524.
17. Grossman, T., Matejka, J., and Fitzmaurice, G. Chronicle: capture, exploration, and playback of document workflow histories. *UIST '10*, ACM Press (2010), 143–152.
18. Gupta, A., Fox, D., Curless, B., and Cohen, M. DuploTrack: a real-time system for authoring and guiding duplo block assembly. *UIST '12*, ACM Press (2012), 389–402.
19. Johnson, J.A. and Nardi, B.A. Creating presentation slides: a study of user preferences for task-specific versus generic application software. *ACM Trans. Computer-Human Interaction 3*, 1 (1996), 38–65.
20. Karrer, T., Weiss, M., Lee, E., and Borchers, J. DRAGON: a direct manipulation interface for frame-accurate in-scene video navigation. *CHI '08*, ACM Press (2008), 247–250.
21. Kong, N., Grossman, T., Hartmann, B., Agrawala, M., and Fitzmaurice, G. Delta: a tool for representing and comparing workflows. *CHI '12*, ACM Press (2012), 1027–1036.
22. Lafreniere, B., Grossman, T., and Fitzmaurice, G. Community enhanced tutorials: improving tutorials with multiple demonstrations. *CHI '13*, ACM Press (2013), 1779–1788.
23. Lee, B., Kazi R. H., and Smith, G. SketchStory: Telling more engaging stories with data through freeform sketching. *IEEE TVCG (InfoVis '13) 19*, 12 (2013), 2416–2425.
24. Lee, E., Wolf, M., and Borchers, J. Improving orchestral conducting systems in public spaces: examining the temporal characteristics and conceptual models of conducting gestures. *CHI '05*, ACM Press (2005), 731–740.
25. Li, Y., Landay, J.A., Guan, Z., Ren, X., and Dai, G. Sketching informal presentations. *ICMI '03*, ACM Press (2003), 234–241.
26. Lichtschlag, L., Karrer, T., and Borchers, J. Fly: a tool to author planar presentations. *CHI '09*, ACM Press (2009), 547–556.
27. Mamykina, L, Mynatt, E., and Terry, M.A. TimeAura: Interfaces for Pacing, *CHI '01*, ACM Press (2001), 144–151.
28. Matthews, T.L. Designing and Evaluating Glanceable Peripheral Displays. PhD thesis, EECS Department, University of California, Berkeley (2007).
29. Nakamura, T. and Igarashi, T. An application-independent system for visualizing user operation history. *UIST '08*, ACM Press (2008), 23.
30. Nguyen, C., Niu, Y., and Liu, F. Direct manipulation video navigation in 3D. *CHI '13*, ACM Press (2013), 1169–1172.
31. Pongnumkul, S., Dontcheva, M., Li, W., Wang, J, and Bourdev, L. Pause-and-play: automatically linking screencast video tutorials with applications. *UIST '11*, ACM Press (2011), 135–144.
32. Sharmin, M., Bergman, L., Lu, J., and Konuru, R. On slide-based contextual cues for presentation reuse. *IUI '12*, ACM Press (2012), 129–138.
33. Spicer, R., Lin, Y.-R., Kelliher, A., and Sundaram, H. NextSlidePlease: Authoring and delivering agile multimedia presentations. *TOMCCAP 8*, 4 (2012), 1–20.
34. Yeh, T., Chang, T.-H., and Miller, R.C. Sikuli: using GUI screenshots for search and automation. *UIST '09*, ACM Press (2009), 183–192.
35. Zongker, D.E. and Salesin, D.H. On creating animated presentations. *SCA '03*: 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation, 2003.