2010-12-16

Capture and Access of Multiple Screen Presentations

Kelv H. Cutler
Brigham Young University - Provo

Capture and Access of Multiple Screen Presentations

Kelv Homer Cutler

A thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Master of Computer Science

Department of Computer Science

Brigham Young University

April 2011

ABSTRACT

Capture and Access of Multiple Screen Presentations

Kelv Homer Cutler

Department of Computer Science

Master of Computer Science

Knowledge transferred during meetings is often ephemeral in nature and thus must be captured if it is to be retained. Ideally, a capture solution should be able to 1) accommodate any number of screens without sacrificing image quality and 2) allow dynamic access to a complete media capture while the capture is taking place. Both students and employees can benefit from the information captured during the lectures and meetings for enhanced discussion and afterward for knowledge retention. Current systems do not support multiple screen capture well, and no system supports dynamic access to the active meeting capture during the meeting. We built a central display server that manages communication to all participants and presenter, manages what is shown on each display, captures all media sent to it and allows playback of that capture on the fly. Static media (images, video, and audio) can be referenced, along with dynamic media (desktop sharing), by any participant's notes in order to initiate and direct playback of the meeting capture – in other words, *rewind* the presentation.

We validated the functionality of our tool by simulating a three screen class lecture where each student performed tasks requiring them to access the capture both during and after the meeting. With basic training, all participants successfully engaged in the rewind interaction and review process.

Keywords: multiple screen presentation, meeting capture, lecture capture

Contents

Chapter 1 – Introduction

Recording information presented during a meeting is an important task that can be accomplished in a variety of ways. There are devices that record audio and video, outlines and presentation files can be distributed, personal notes can be taken, etc. Ideally, a capture solution should be able to 1) accommodate any number of screens without sacrificing image quality and 2) allow dynamic access to a complete media capture while the meeting is taking place. Numerous meeting capture tools exist (see chapter 2); however, nearly all tools are limited to two screens and require the capture to be completed before referencing the data in the capture. Capture tools typically have a finalizing step where the capture is processed and packaged up for distribution. Consequently, no one attempts to use what is actively being captured because it would mean: 1) the capture is halted for a period of time, and 2) the resulting capture is then segmented into two separate captures. For dynamic access of the active meeting capture to even be considered, the capture process must be re-evaluated.

Two areas, in particular, need better support: multiple screen presentation capture and accessibility of the capture during the meeting. With multiple screen presentation support, participants and reviewers will have more contextual information available to them as they learn. However, converting all media into a compressed video must be avoided because it will degrade the image quality available to the reviewer. Without the ability to access the capture as it is being captured (the “active capture”), referencing material presented earlier in the meeting is not supported. Our solution, which addresses these two problems, allows participants to view more context (multiple screens) and full detail (lossless capture) during review and has the potential to enhance the learning experience during the meeting (through dynamic access to the active capture).

Multiple Screen Presentations

Multiple screen presentations allow a presenter to reference and explore a variety of information sources without losing context. For example, a computer science professor may demonstrate how to implement a simple project during class. Ideally, the professor would display the program specifications, a reference sheet for API's in applicable libraries, and the editor he is using to write the code all at the same time. The context (program specs) and supplementary information (reference page) cannot co-exist on a single screen with the main content (editor application) without adding more screen space (see Figure 1).

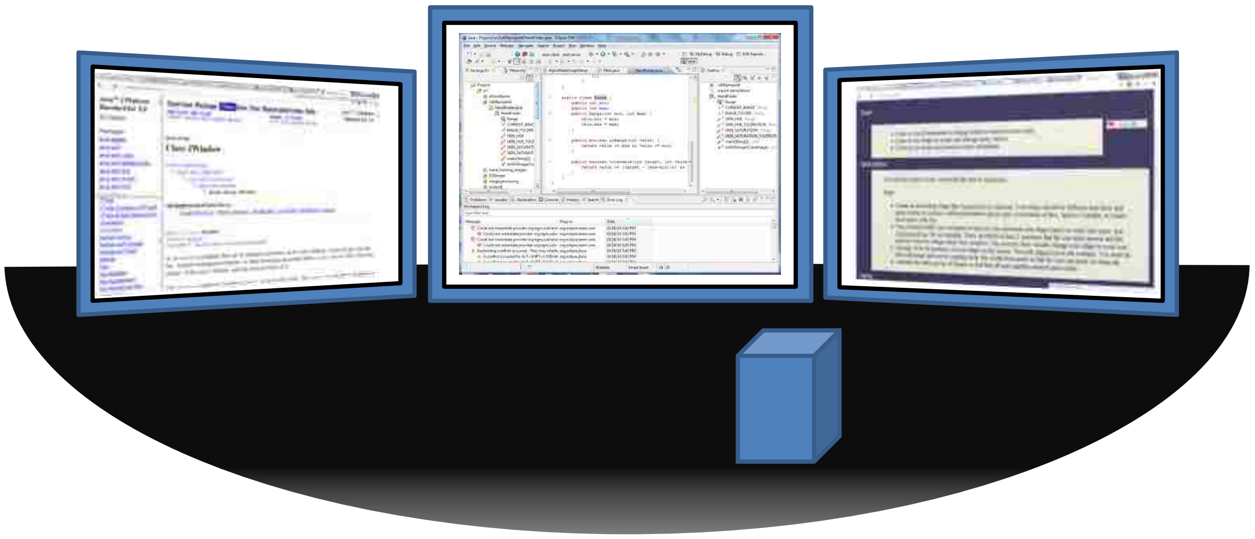


Figure 1 – An example of a multiple screen presentation. Left: Java's API for JWindow. Center: Eclipse code editor. Right: Program specifications.

Math teachers can make similar use of multiple screens. A persistent view of the outline would allow the students to see the big picture of how each lemma and piece of a given theorem fit together. Applicable formulas could fill another screen while another couple of screens could be dedicated to proof formulation or working through examples as a class. Many other disciplines and meeting scenarios could benefit from added screen space, all of which would need to be recorded in a capture system.

A number of capture systems that attempt to support multiple screens resort to rendering each screen into a separate video file. Doing so will constrain all media to a fixed size/resolution and will often degrade image quality further by applying a lossy compression codec to the video file. Such reductions in the image quality will not allow reviewers to access the original detail presented during the meeting – small text and complex diagrams will be unreadable. For example, in Figure 1, all three screens contain relatively small text and detailed information. When reviewing, context may be a priority and so all the screens may be shrunk to fit on the one screen (screen A of Figure 2). In this situation, a loss in image quality would not be noticed. However, when trying to decipher smaller text or points from a complex diagram, detail and maximum resolution become the highest priority. In this situation, a single screen may be enlarged (screen B of Figure 2) and poor image quality will prevent the reviewer from getting information out of the capture. Thus, it is critical to preserve media in its original form.

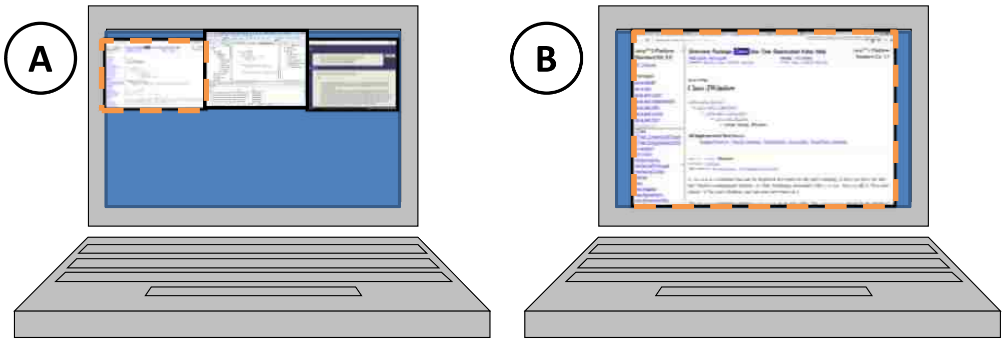


Figure 2 – Fitting a multiple screen presentation on a personal single screen can be a relatively easy task. However, there is a tradeoff between keeping context (screen A) and maintaining sufficient resolution screen B).

When reviewing a multiple screen presentation on a single screen, both context and detail are not possible simultaneously – there must be a tradeoff. One goal of our solution is to allow the reviewer access to both. Original detail and resolution cannot be reproduced if image quality has been sacrificed by converting the capture into video format. The solution to supporting full detail in multiple screen capture and review is found in the technique used for meeting capture (see Chapter 3), and the interaction provided to the reviewer (Chapter 8).

Accessing the Active Presentation Capture

The other major goal of this research project is to demonstrate a viable way of referencing prior material in the meeting capture or, in other words, *rewinding* the presentation. Although there are many ways to reference the meeting capture, this solution focuses on using participant's notes as the reference points. With the ability to reference the meeting capture via notes, participants can direct a rewind of the meeting capture to the place at which a specific note was taken. Doing so then allows everyone in the room to view the material presented previously relevant to the individual's note.

For example, a biology teacher is lecturing on DNA and the life-cycle of a cell. A student may have a question about the assembly of mRNA. On the presenter's screens, there is the lecture outline, a diagram showing DNA transcription, and a number of bullet points about mRNA (see Figure 3). The student has written in his notes that DNA is unzipped before transcription step 1 of Figure 3. Much later in the presentation when he sees a polypeptide chain assembled via tRNA, he thinks back to mRNA and wants to know how pieces of mRNA are assembled. Because the presenter has moved on to a different topic, he, and likely the rest of the class, may have difficulty switching context when the question is posed: "is there a molecule like tRNA that pieces the strand of mRNA together?" The complex diagrams used to illustrate the DNA transcription process along with bullet point details about mRNA shown earlier would quickly bring the entire class to the same page. The student with the question can reference his notes – step 2 of Figure 3 – in order to direct the access of the meeting capture – step 3 of Figure 3.

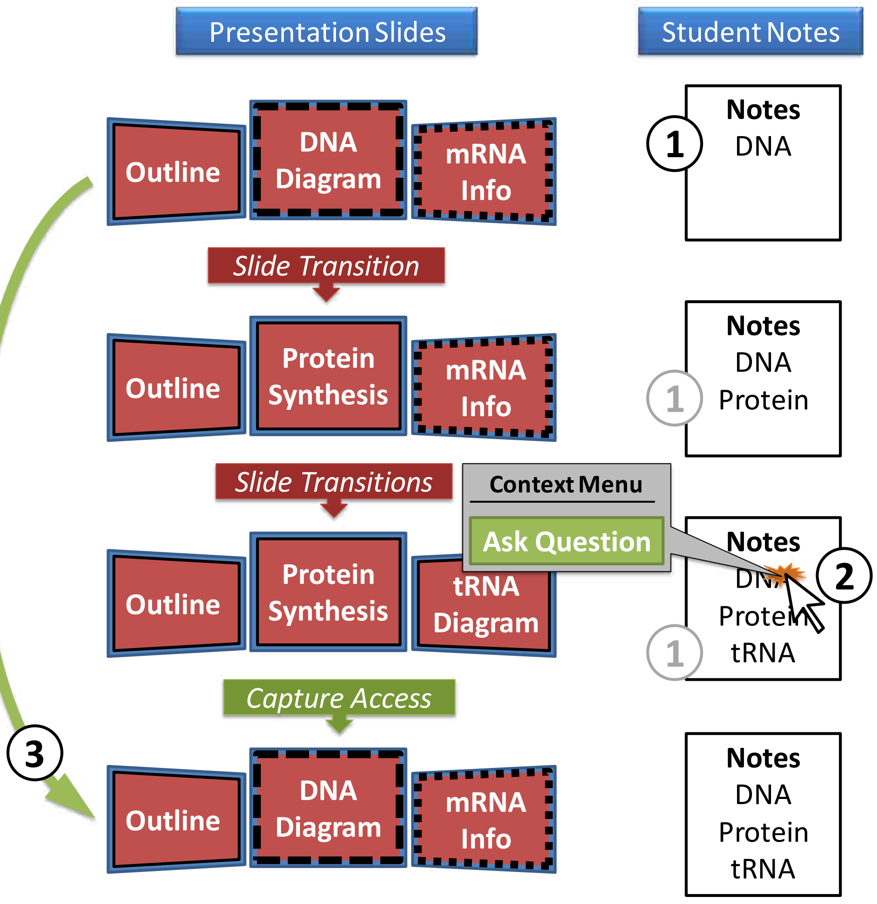


Figure 3 – 1) Notes are taken at various places in the presentation. 2) A note that is "linked" to a previous point in the lecture capture is referenced during the note-taker's question. 3) The corresponding material in the meeting capture is accessed and displayed to aid the presenter in answering the question.

Although rewinding a presentation could occur simply by navigating to the appropiate slide, there are many situations where this is not possible. For example, a math teacher may demonstrate how to use MATLAB in class. The teacher may generate a number of graphs. A slide based presentation and meeting capture system would not record the use of MATLAB. To support rewind and full capture access during the meeting, the system must capture unknown forms of media (MATLAB) in addition to the standard forms of media (images, movies, audio, etc.). Students in the class may struggle to accurately describe their questions about MATLAB because it is a complex piece of software. Therefore, it is critical for any capture system to capture all forms of media and allow students to refence any piece of the capture.

Allowing dynamic access to the presentation capture during the meeting without interrupting the capture is not a trivial problem. A handful of solutions do provide remote viewers immediate access to the meeting capture (transmitting a video feed similar to video conferencing); however, not being built with the goal of local – in class – access, none of them support referencing the capture or pulling up the material from the capture on the active presentation displays. Thus, we distinguish the ideal interaction for accessing the capture as *dynamic* access versus *immediate* access – which does not keep the capture accessible and linkable on the presentation displays during the presentation.

A number of techniques exist that could be used to solve dynamic access to a meeting capture. This paper presents a solution that produces a capture which is immediately available for access on the main presentation screens and can be referenced or navigated in a variety of ways. Chapters 3 and 5 discuss the capture technique used in this solution. Chapter 6 demonstrates unique playback controls afforded by the capture technique chosen. Chapter 7 discusses a few of the issues encountered when the ability to rewind the presentation is given to participants.

Thesis Statement

Presentation capture systems should include flexibility for multiple screen presentations and dynamic access of the active capture.

A solution that accomplishes the goals described in the above thesis statement will have the following properties: 1) multiple (more than two) screens can be captured and reviewed simultaneously, 2) captured media can be accessed with equivalent quality on a single small display device (as might be found on a laptop) as the original media displayed on one or more large projection screens as used in the meeting, 3) all forms of media can be captured, and 4) referencing and accessing capture material during the meeting occurs without interrupting the active capture (dynamic access).

In Chapter 2, related works are evaluated according to the objectives listed above. Chapter 3 gives an overview of our solution and the enabling technologies that were either integrated or created. In chapters 4 through 8 the major challenges faced in implementing the solution are laid out along with the solutions and ideas that were or could be used to solve each challenge. The resultant solution was evaluated by ten participants in a task based user study. The tasks required participants to use the note taking and reviewing applications to demonstrate the viability of dynamic capture access and to observe interaction behaviors with the software. The conclusions about viability and the observations made are presented in chapters 9 and 10.

Chapter 2 - Prior Work

This chapter categorizes existing capture systems into the following five classes: Online Meeting Services, Standard Lecture Capture, Slide and Annotation Based Lecture Capture, Dynamic Access of Meeting Capture, and Flexible Multiple Screen Support. Throughout all of the related solutions, screen support appears to be limited to, at most, two screens with the exception of Qumu

7]. Referencing material from a meeting capture, or using participant's notes to navigate the capture, is not new; however, systems providing this feature do so only after the meetings have ended [13, 20, 2, 6, 8]. There are a handful of systems [11, 15, 18, 19] that allow presenter-participant interaction with presentation content; however, none of those systems attempt to allow access to a full scale meeting capture – just slides and annotations.

Online Meeting Services

Many systems that facilitate online meetings for collaboration also provide meeting capture. Systems in this category support only one main screen or input. TeamSpace [21], WebEx [10], DimDim [1], and GoToMeeting [3] are a few prominent examples of the "online meeting services" class of capture system where, as a meeting occurs online, a recording of the shared display is provided in a video-like format for later playback. We have not listed all such tools, but have given a representative sample. The presenter may choose what is shown in the main display space (e.g. a shared application, presentation file, or collaborative whiteboard). Capture support in these systems cover – for the most part – all forms of media, but do not allow live access to the capture. None of these systems deal with multiple screens since there is only one shared space in which to work. Most of these commercial meeting products hide their implementation so whether the full resolution and quality is preserved or not is unknown.

Standard Lecture Capture

There is a plethora of lecture capture systems ranging from commercial to open source to homegrown projects. Echo360 [2], Panopto [6], and Tegrity qumu.com, 2010.

8] are three of the larger commercial products. They all record lectures and present them for review in a web browser and a number of other media formats. Opencast [5] has an open source project called Matterhorn built to facilitate lecture capture. In each of these systems, the presenter’s computer screen and webcam image may be captured and played back in a video-like format with various indexes into the video usually based on slide transitions. Tegrity and Panopto currently support linking student notes with the presentation capture, but the linking functionality is only available during personal review. Echo360 and Matterhorn do not claim to do linked notes or live access of an active presentation capture. Similar to online meeting spaces, the main display can easily switch between a camera, virtual whiteboard, or desktop capture. This feature is most similar to having multiple screens but does not bring with it the same power of retaining full context. All of these systems including many homegrown projects appear to be rigidly set at a maximum of 2 screens. Tegrity and Panopto appear to preserve most if not all the quality of the presented media, while there is definitely a loss in quality with Echo360 and Matterhorn. Many more lecture capture systems exist than the four discussed here, but were not mentioned because the support they offer similar to the discussion above.

Slide and Annotation Based Lecture Capture

Other capture tools take a slide image approach rather than a video playback approach. ProjectorBox [17] is a system that records a stream of pixels as they are transmitted to a projector. ProjectorBox segments the image stream into slides, processes each image and allows search over the text in the images. Although the system can process and publish the images during the meeting, notes are not linked to those images and captured media is limited to images. Aside from browsing the images with additional post-processed information, ProjectorBox does not contain additional features related to the work in this thesis.

Campus Mobile [16], Classroom Presenter [11,12], Classroom 2000 [13], and Authoring on the Fly [19] are systems that distribute a set of slides initially or as needed, and then record slide transitions and slide annotations made during the meeting. Thus a full capture includes the slide images in their original form along with a log of transitions and annotations. Classroom 2000 is built to record additional streams on top of the transitions and annotations such as URL visits and video derived gestures from the presenter. However, even with Classroom 2000 the number of main displays available for review is limited to one. Three of the four representative systems for slide based lecture systems do provide interesting interaction with presentation material, but does not provide access to a full capture. For example, Campus Mobile and Authoring on the Fly synchronize all slide transitions and annotations with every person connected to the system as they occur. Classroom Presenter has special “activity slides” where students can submit their slide annotations to the presenter.

NoteLook [14] provides multiple visual streams available for capture – slide images or video feeds – but it relies on the viewer to explicitly take a snapshot during note-taking. NoteLook then inserts each snapshot into the student’s notes with a flow layout. ModSlideShow [15] used the NoteLook software to allow capture and annotation of slides, but adds the ability to send a slide image with annotations back to a shared display for all to view. ModSlideShow does not inherently provide full access to the presentation capture; it is limited to whichever media a student chooses to capture. Complete capture access via the student’s notes could be achieved if only images were used and the student actively each one, but this would severely limit the forms of media that can be used.

Dynamic Access of Meeting Capture

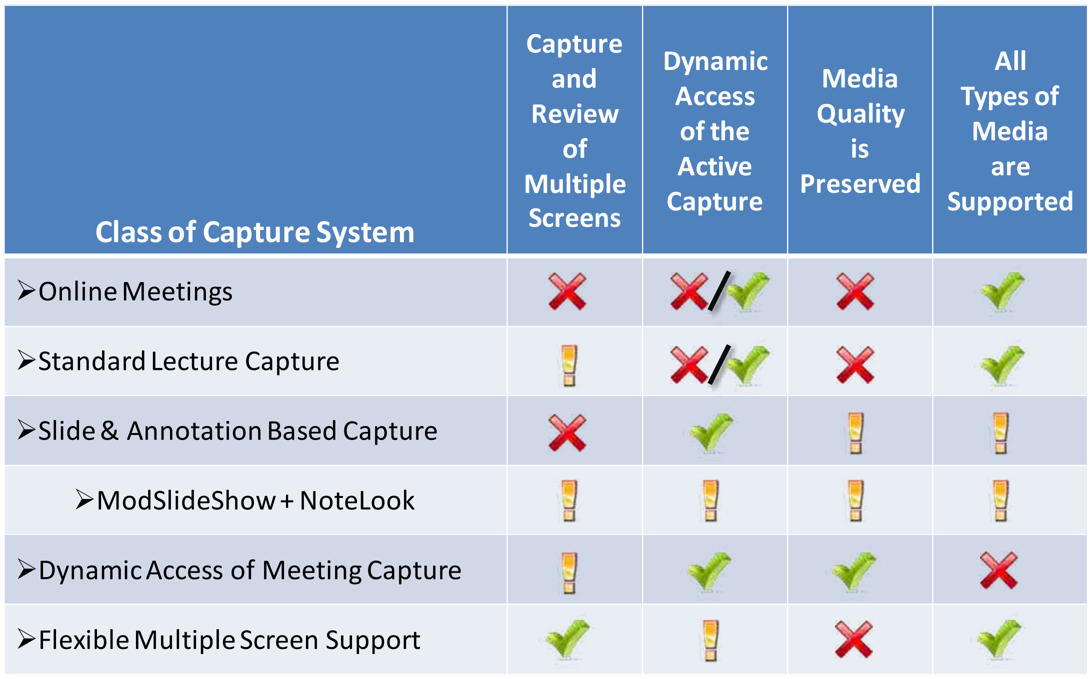
TideBreak [9] has developed a system for meeting collaboration with file sharing and input redirection at its core. After files have been shared with a specific machine – or all machines – a participant's cursor can move onto a shared/public machine such that multiple participants can interact remotely via their personal keyboard and mouse. The capture output from a TideBreak meeting is a history of all the files shared between machines. No other visual, audio, or interactive data is captured. However, TideBreak does allow access to captured content live during a meeting. TideBreak asserts that the live access of the capture is a highly useful function of its system. Unfortunately, although media is preserved in its original form and could have appeared on any number of screens, the amount of information captured is not enough to claim all forms of media (missing desktop interactions) or multiple screens.

Flexible Multiple Screen Support

Qumu [7] is a commercial lecture capture system that allows up to 4 feeds of video to be recorded at once and played back in a flexible manner. Flexible refers to the ability to move and resize multiple screens. This system is one of very few that provides capture of multiple main displays and the only one to offer such flexible control over the multiple screens during review. Flexibility with multiple screens is important because it allows the reviewer to re-balance the tradeoff between viewing context and detail as needed. Qumu can handle any form of media, but does not provide any linking to notes or access to the active capture during the meeting. Multiple video feeds (up to 4), can be seen during review, but all forms of media are reduced to a compressed video format such that there is usually a loss in quality.

Summary

The table shown below in Figure 4 provides a quick visual for where each class of capture system succeeds or fails in accomplishing the objectives described in chatper 1. ModSlideshow + NoteLook is one system that provides unique dynamic access. Consequently, it was considered explicitly in the table below.



Class of Capture System	Capture and Review of Multiple Screens	Dynamic Access of the Active Capture	Media Quality is Preserved	All Types of Media are Supported
➢Online Meetings	✗	✗/✓	✗	✓
➢Standard Lecture Capture	!	✗/✓	✗	✓
➢Slide & Annotation Based Capture	✗	✓	!	!
➢ModSlideShow + NoteLook	!	!	!	!
➢Dynamic Access of Meeting Capture	!	✓	✓	✗
➢Flexible Multiple Screen Support	✓	!	✗	✓

Figure 4 – Summary table showing each class of capture system and how it does or does not fulfill each objective. Legend: ✗ No support for the objective, ! Partial fulfillment, ✓ Meets objective, ✗/✓ Some systems in the class of capture seem meet the objective while others do not

Multiple screen support during capture and review is supported well by Qumu, and only in a limited form with standard lecture capture. TideBreak and NoteLook support capturing content from multiple screens, but do not visualize the multiple screens during playback. Whether a system preserved full image quality or not appears to vary throughout and within each class of capture system. Access to presentation content during the meeting is supported in slide based capture systems and TideBreak allows complete access to what was captured. However, both slide based capture systems and TideBreak do not capture interactive forms of media (e.g. playing a movie/audio file, desktop interaction). All other classes of systems do support capturing any form of media.

Chapter 3 – Solution Overview

In this chapter, we briefly introduce the method of capture and pre-existing framework used to implement our solution. Next, the framework used to support the solution is described. Finally, the key challenges faced in building the solution are outlined with references to their corresponding chapters.

Capture Method and Existing Tools

The method used to capture a presentation is crucial to what features and interactions the entire system can support. For example, if the chosen capture method is simply to record everything via video camera(s), then many features cannot be supported. From the camera's recording, the true form and quality of the media is not recoverable. Additionally, accessing the camera's recording in order to review something recorded earlier is possible but it is inconvenient and will not support live access to the tape without interruption. Therefore, the capture method must be chosen carefully.

According to the objectives set forth in the introduction, a capture system would need to support the following features: multiple screens in capture and review, lossless image capture, dynamically accessible capture, and support for all media types. Architecting a solution that includes all of these features could be done in one of three ways: 1) *video*: continually build and encode a video file of what is shown, 2) *plug-in*: "plug in" to existing presentation software, such as PowerPoint, and record transition events, and 3) *from scratch*: build an entirely new system which manages the display and archiving of all media to all available screens. The following paragraphs discuss trade-offs for each method.

1) video: Continually building a video file from screen capture is a straightforward approach. However, we are not aware of a codec that compresses efficiently enough to handle multiple screens at once without violating one of the core objectives: 1) lossless image quality and 2) playback of the video while the video is actively being encoded. Video rendering and compression is a processor intensive task. According to Wikipedia [4], a typical high definition video encoding requires between 3 - 5 Mbit/s bandwidth to be streamed. Therefore, the processor must process and encode over 3 - 5 megabits of data per second per screen. Supporting this task with specialized hardware may be possible and something to consider. However, without deep exploration, this approach did not appear to be feasible without sacrificing image quality and was not pursued.

2) plug-in: Using a presentation tool like PowerPoint would allow capture and replay to be a simple process. The strategy is simply to record each new state of the presentation and distribute the presentation file along with a log of the different states. A single *state* of the presentation would include things like: slide index, bullet point index, cursor position, and corresponding timestamp. This solution is great for simplicity when the presentation consists of static media; however, a significant amount of effort would be needed in order to make software demonstrations capture-able, accessible, and distributable. Such effort would likely be comparable to starting from scratch like method 3 suggests. The only reason to consider this method would be to preserve familiarity with the presentation tool of choice.

3) from scratch: Creating a centralized display server application that receives media and renders it to whichever screen is specified makes the capture process straightforward. The display server simply needs to log and archive all media received and where it was shown. This will produce a lossless capture that can be accessed at any point in time. To replay the capture, the display server simply needs to traverse the log to find which media was displayed where at the specified time. If the presentation includes a software demonstration, such as typing up a computer program during class, then the stream of screen captures sent to the display server can be logged away just like any other media. With this technique for capturing a presentation, both goals of multiple screens and dynamic access can be realized without any loss in image quality.

The “from scratch” method for capture was chosen for the solution described in this paper. The display server application mentioned in the “from scratch” method was built in part as a separate project in the BYU ICE lab. This system, known as SPICE, became the foundation and framework for presenting, capturing, and all interaction having to do with dynamic capture access.

Use of SPICE in the system

SPICE has been developed in the BYU ICE lab for a variety of research purposes. The SPICE system allows multiple mobile devices to connect with a display device and share windows (applications), desktop, or other forms of media. The display device runs a server process which manages connections to each mobile device and renders the display commands. As shown in the diagram below (Figure 5), a user can choose to share various forms of media (#1) on the display (#3) and does so by transmitting display commands (#2).

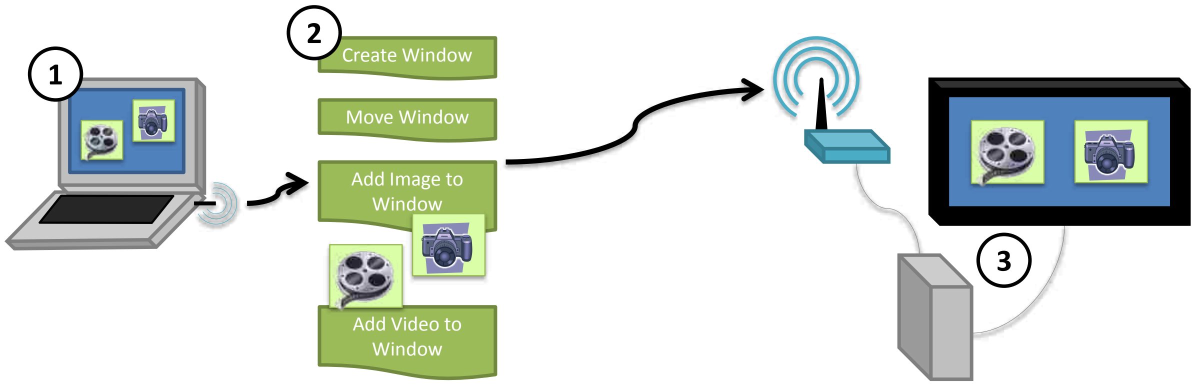


Figure 5 – Mobile devices can choose windows or media (#1) to display on the device it is connected to (#3) by sending display commands to the display device (#2).

Although many of the essential pieces to the solution existed in the SPICE system, no client software to take notes or to build and control a multiple screen presentations existed. The original system also lacked support for capturing and archiving all media on the server side. SPICE was a good foundation to build on, but needed major modifications and additions to support presenting, note-taking and meeting capture.

Modifications to SPICE

Adding capture to SPICE was fairly straightforward. The display server process was modified to log all display commands received and archive all media used during the presentation. The display server process was also modified to include audio recording if a microphone is connected to the machine. Figure 6 shows these two key modifications made to the SPICE system.

In step 1 of Figure 6, mobile devices running client software have media content they will share with the display server. Display commands are sent across the network to the display server along with the necessary media files in step 2. The display server, shown in step 3, is a combination of display device(s) and display server process. It receives all display commands and media content. The server process renders the content on the display(s). Finally with the modifications to the display server, it then archives all display commands and media content into the capture archive.

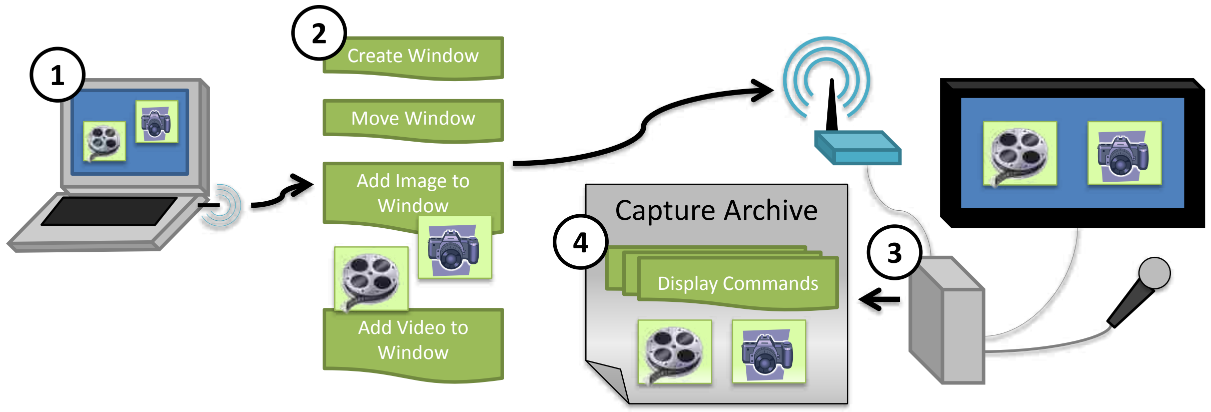


Figure 6 – Architecture for capturing a meeting. 1) Client software with presentation software. 2) Display commands which dictate where to show various media elements. 3) Display server. 4) Capture archive which holds both a log of display commands and media files.

This style of logging display commands leads to dependencies between a display command and all other display commands previous to it. Full details on the dependencies brought about by this method of capture and the solutions to them are discussed in chapter 5.

SPICE was originally built to facilitate window management. It, therefore, allows a single machine to be in control of the displays by approving or denying each display command coming from any number of machines. This portion of the framework fit nicely with the scenario of a meeting attendee requesting approval to access the meeting capture. Display commands that control access of the capture and an interface to approve or deny the display commands needed to be built. Once built, anyone can access and control the meeting capture – provided the action is approved by the machine in charge (a.k.a. presenter). This interaction between participant and presenter is modeled in Figure 7.

As shown in Figure 7, the user interface provides a context menu allowing the user to ask a question about a specific note. The "ask a question" action sends a rewind request display command to the display server as seen in step 1. In step 2, the display server sends a message to the presenter indicating that a display command requires a decision to be made. Once the presenter makes a decision, it sends its response to the display server, see step 3. The display server then acts accordingly by either ignoring the original display command or by accessing the locally held capture to "rewind" the presentation.

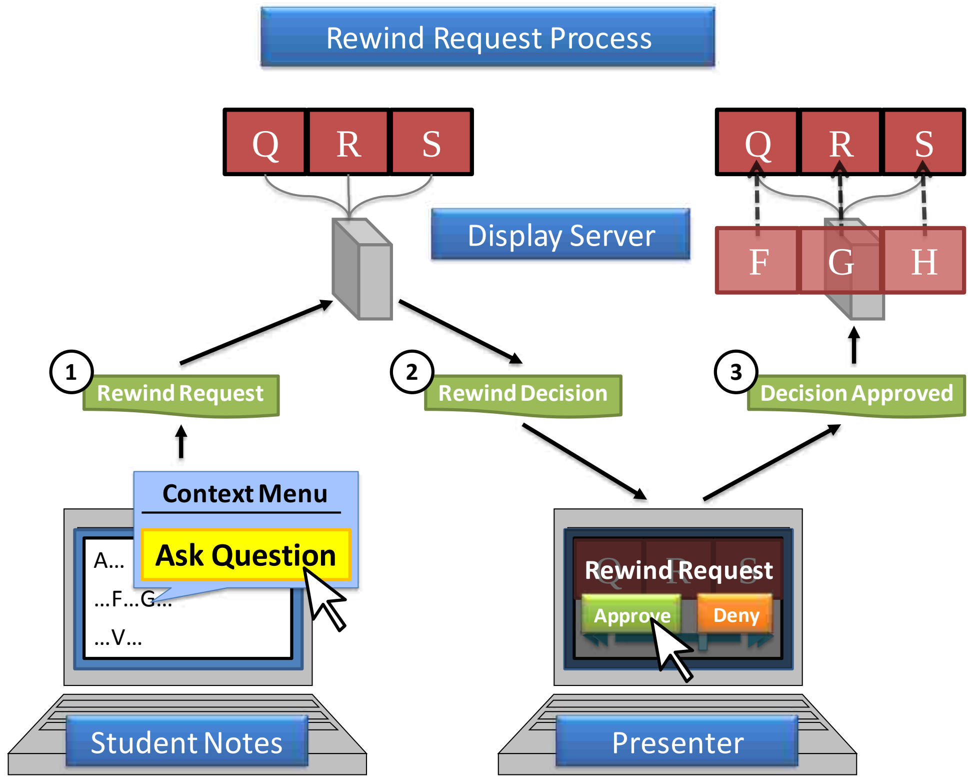


Figure 7 – 1) Personal notes with context menu for requesting a rewind. 2) Message given to the presenter asking them to make a decision. 3) Message from presenter to display server.

The interaction between a participant and the presenter has some potential pitfalls. Allowing meeting attendees to request a rewind of the presentation opens up the potential for unwanted disturbances. Although not part of the core objectives, managing the interactions dealing with dynamic access was an important piece to this solution. Therefore, these problems and possible solutions are discussed in chapter 7.

Adding Presentation Support to SPICE

SPICE did not include any presentation or note-taking software. Note taking software has been well explored so only a simple note taking application was built. Building presentation software to handle multiple screen presentations and capture playback is a difficult task that needs additional research time. Our solution provides a basic set of features necessary for allowing a user to present on multiple screens and control a meeting capture. While it is a sound solution with interesting ideas worth discussing, we do not claim to have found the ideal way to author and show a presentation.

Returning to the scenario of a computer science professor presenting on multiple screens will reveal basic features that the system should support. The professor plans to start the lecture with some slides on object inheritance and a video further explaining the idea. He wants some of the slides to appear at the same time on different screens. Later he wants the slide with the specification for a programming assignment to show on the left-most screen while he opens up an editor on the main screen and the Java API home page on the right-most screen. When the program is built and run, the professor will want to show the output screen over top of the program specs. From this example we can see it is important to be able to use and organize many forms of media and software demonstrations in a logical sequence. The screen on which each presentation element will be shown is usually predetermined. However, adding presentation elements or adjusting where they are shown should not be precluded.

Each of the basic features mentioned the above paragraph are included in this solution. Details on how a presentation is assembled in this solution can be found in chapter 4. In addition to presenting, an interface for accessing the active capture was built with following control options: play/pause, timeline based navigation, navigation via linked notes, event (e.g. slide transition) based navigation, and request/approve controls for linked notes. Playback controls are an important part of the user experience. Therefore, chapter 6 was written to discuss the implementation and rationale for playback controls. These controls affect both live – during meeting – playback found in chapter 7, as well as personal review playback discussed in chapter 8. Additional issues discussed in chapter 8 include screen resizing and the general architecture used for the reviewer software.

Chapter 4 – Building the Presentation Framework

The solution presented in this paper supports three roles for participating in the meeting: Author, Presenter, and Note-Taker. The Author role allows a user to build a multiple screen presentation and include any form of media including software demonstrations. Showing a presentation, or the Presenter role, is where a user can present a pre-built presentation and manage dynamic capture access, including participant's rewind requests. Note-Taker is the role users take as participants in a meeting where they can take notes, request a rewind, and, at a later time, review a meeting capture.

Presentation Authoring

When creating a multi-screen presentation there are a number of tasks involved. We broke it down into three groups: setting up the presentation, putting media in sequence, and then assigning windows for each media element. The presentation file is actually a folder with all the media collected together along with the data files used for organizing it. Choosing where to save the presentation is the very first step.

In setup phase (see Figure 8), the user must decide how many *windows* – or media placeholders – are desired; this number will often correlate directly to the number of screens expected to be available during the presentation. Windows can be added and positioned to fill an entire screen, fill a part of the screen or split a screen with another window (see items 2 and 3 in Figure 8). Positioning windows at this point is optional. Adjusting windows is a necessary step when the target screen configuration is known. The screens will be represented in the window arranger when connected to a display server or when a screen configuration is selected from file (see item 4 in Figure 8). Various window layouts can be saved and selected in the drop-down seen in

Figure 8 item 1. Item 5 in Figure 8 is where software demonstrations can be declared for later use in the presentation.

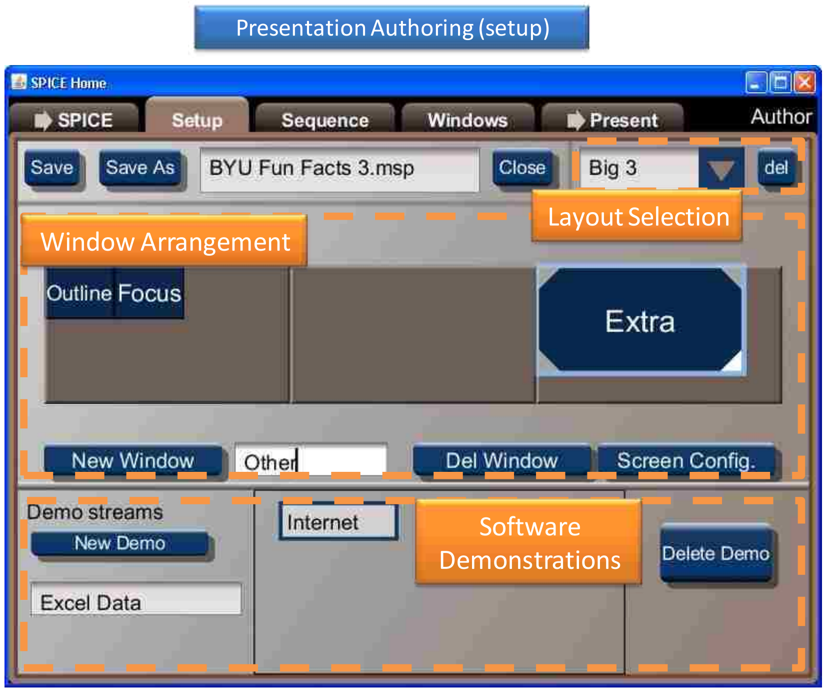


Figure 8 – Layout Selection lists previously saved window layouts. Window Arrangement allows the windows to be adjusted to fit various screen configurations. Software Demonstrations can be specified ahead of time.

In the next phase, “Sequence” (Figure 9), media is imported into the “pool” and then positioned as desired in the presentation sequence. Once drag and dropped into the sequence, each media element is considered a *slide* and can take on various properties. One of these properties is the notes that are specific to each slide. The window identifier is another important attribute of the slide – it determines which window will host the media. A slide may need to appear at precisely the same time as the previous one or two slides. Grouping slides such that they will appear simultaneously is called “coupling” slides in this presentation software. Therefore, “coupled to next” is another property of a slide.

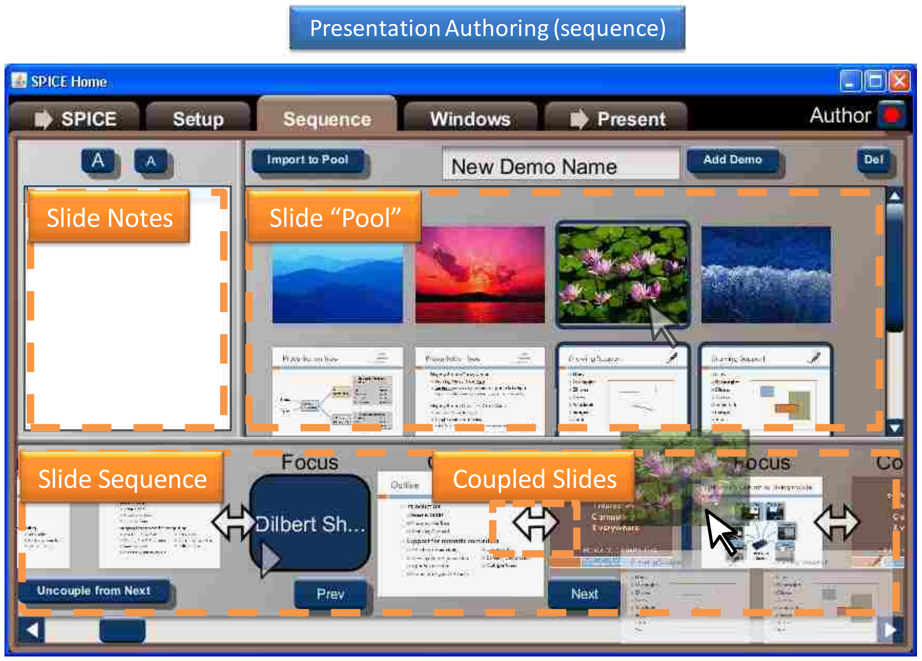


Figure 9 – Interface for assembling a multiple screen presentation.

Lastly, each slide needs to be assigned a window to appear on or else it will be skipped in the presentation. The name of the window it will appear on is found just above the slide. The window name can be set via the context menu (shown in Figure 10), or in the next phase of presentation authoring, "Windows." The user interface for "Windows" simply gives a preview of where and how each slide's media will appear and allows the window property to be set by a click action on the desired window.

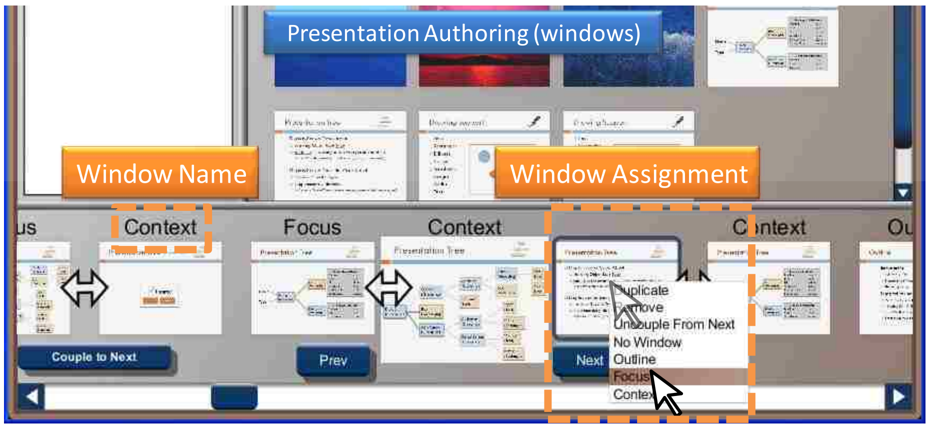


Figure 10 – Each slide or presentation element needs a window assignment.

Showing a Presentation

Once a presentation has been authored and the user is ready to present, they enter the “Presenter” role in this software. The software first directs them to connect to the target display server they will be presenting on. The connection is made by entering the IP address or URL displayed prominently on the display device(s).

After the connection is made, the user is taken to the pre-show setup (shown in Figure 11). As part of pre-show setup, the windows can be placed and adjusted on the available screens, or a pre-configured layout can be selected from the drop-down. The other setup requisite to getting started is identifying all planned demonstrations and specifying what will be shared during the presentation. The user can specify a single window, a portion of their desktop (region), or their entire desktop. After pressing the “window” button the user is prompted to click on the window they want to share during the presentation. If the “region” button is pressed, the user can draw out a rectangular area specifying the region of their desktop to share. The “full screen” button links the demo such that their entire screen will be shown.

When pre-show setup is complete, moving to the “Show” tab will allow the user to start presenting (also shown in Figure 11). “Start Archiving” is the button that controls whether or not the meeting will be captured or not. Slide specific notes can aid the presenter. In the main panel of the Show tab, the screens, the windows, and a thumbnail provide a visual for what is currently being displayed on the display server.

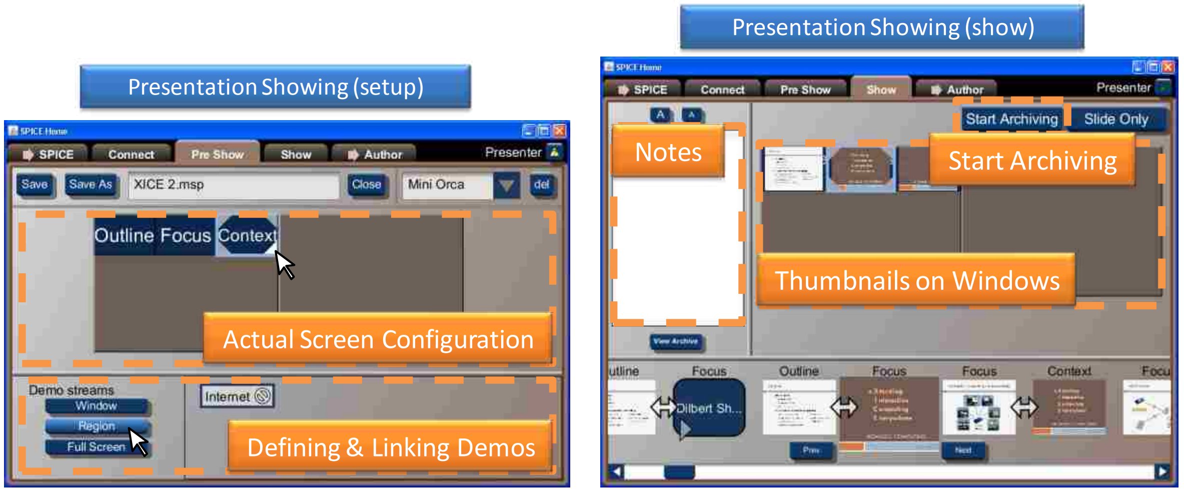


Figure 11 – Pre Show and Show screen shots show various presenter options.

Note-Taking

Users who wish to take notes during the presentation can open up the note-taking application. The interface for this piece of software is simple. Users connect with the display server, entering in their name and the IP address or URL of the display server (Figure 12). This connection information is used later when the user requests a rewind. The rewind request gets forwarded from the display server to the presenter who can then make the decision to rewind based on who and where the request is for. The connection to the display server is also used for two other purposes: 1) Capture Identification and 2) Time Difference Resolution.

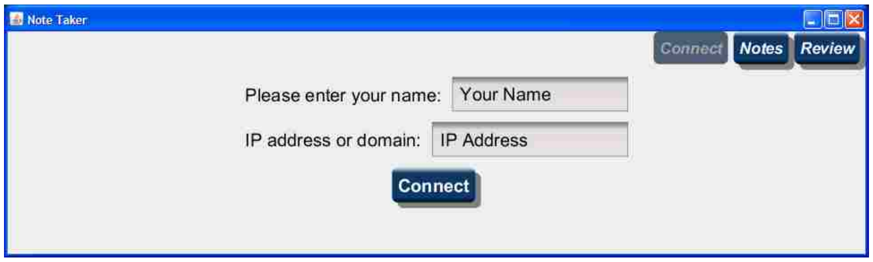


Figure 12 – The “establish a connection” screen.

Each meeting capture has an identifier that aids note-takers and reviewers in linking their notes with the correct capture archive. When the display server begins capturing, it communicates the capture identifier to all connected machines. Or, if the display server has begun capturing before a client connects, the capture identifier will be contained in the initial server description received upon successful connection. This identifier is stored away in the notes file for later use. During personal review, the capture identifier allows the software to identify and locate the meeting capture files (both media files and log information).

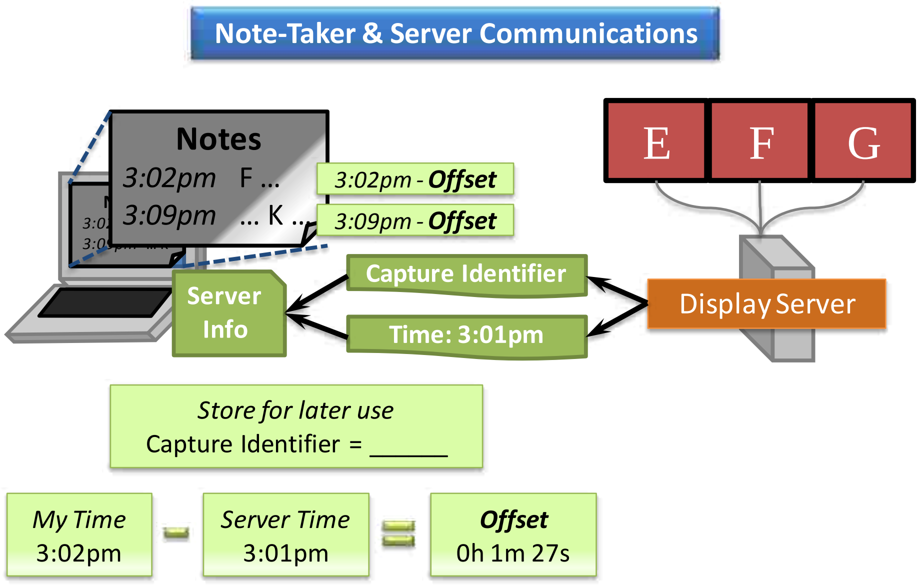
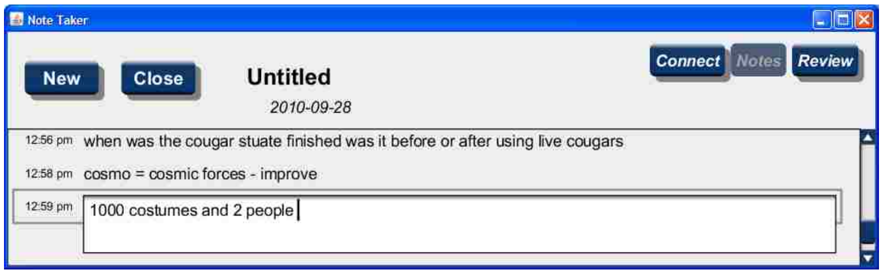


Figure 13 – Server state being communicated to the client note-taking device.

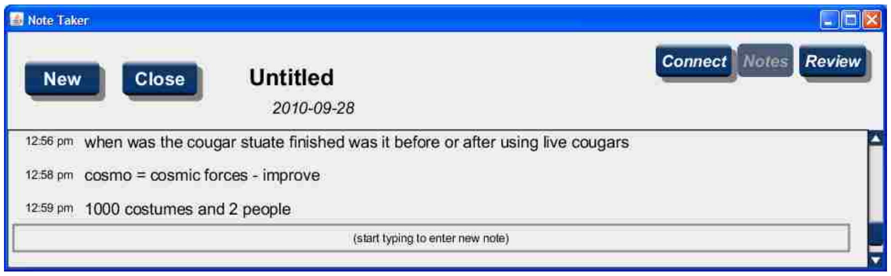
If notes are to be used as reference points into the meeting capture, then the time stamps for each note taken on a student's machine must match the time stamp for the corresponding log events on the display server. This will not be the case if the two computers have conflicting times. Whether the time was adjusted to match a student's personal clock or the presenter traveled in from a different time zone, machine times can vary from a few seconds to a few minutes to a few hours. A matter of a few seconds is insignificant. However, a few minutes or a few hours can mean missing the reference point by a few slides or missing the meeting entirely.

Reconciling these two times is important for capture access both during the meeting and during personal review. During the meeting, the note taking application will make requests to the display server based on timestamps created in the note taker's native time plus some offset. After a meeting, the software will match server recorded timestamps in the display log with individual note timestamps created according to the note taker's native time plus the same offset. As seen in Figure 13 the server time is received, then used to calculate the offset and then stored inside the notes file.

In the note-taker application, notes do not flow into each other, but are purposefully segmented. Each time the user presses "Enter", presses "Escape", or clicks away from the active text area, typing again will start a new note. Clicking on the next "line" will also start a new note. In this manner, notes can be identified clearly with a single, unambiguous timestamp. If desired, notes can be edited at any time, but only the creation timestamp is used in capture access.



12:56 pm when was the cougar stuate finished was it before or after using live cougars
12:58 pm cosmo = cosmic forces - improve
12:59 pm 1000 costumes and 2 people



12:56 pm when was the cougar stuate finished was it before or after using live cougars
12:58 pm cosmo = cosmic forces - improve
12:59 pm 1000 costumes and 2 people
(start typing to enter new note)

Figure 14 – Note taking user interface.

Creating roles for authoring, presenting and note taking were the first steps towards the overall solution. The remaining challenges were found in supporting meeting capture and dynamic access of the meeting capture.

Chapter 5 – Display Logs

In order to preserve quality and maintain accessibility during the capture, we chose the technique of capturing all media used by the display server as *display events* into a log file. Although there are many benefits in using this technique, as mentioned in chapter 3, using a display log also comes with a number of difficulties that must be remedied. The first difficulty is the need to traverse all display events previous to the desired position in the log in order to accurately produce the same display. This can be time consuming and processor intensive. Avoiding such inefficient log traversal can be done by the use of *key frames*. The second difficulty comes with types of media that have separate playback controls (e.g. video and audio). The solution is to track playable media and intercept certain display events before replaying them to the review display server.

Key Frames

The first difficulty is that a log of events makes each event dependent upon the events leading up to any particular display event. This can be seen in the following simple example (see Figure 15). The original sequence is shown first, and then two traversals of the log are shown. The first traversal attempts to start at 3:05 pm which results in error. The error occurs because the ginger bread man picture cannot be placed on a window that was never created (window A). The second example illustrates what can happen if even one event in the log is skipped. There is the potential for errors again, lost content, or content that shouldn’t be there (happens when a remove media event was skipped).

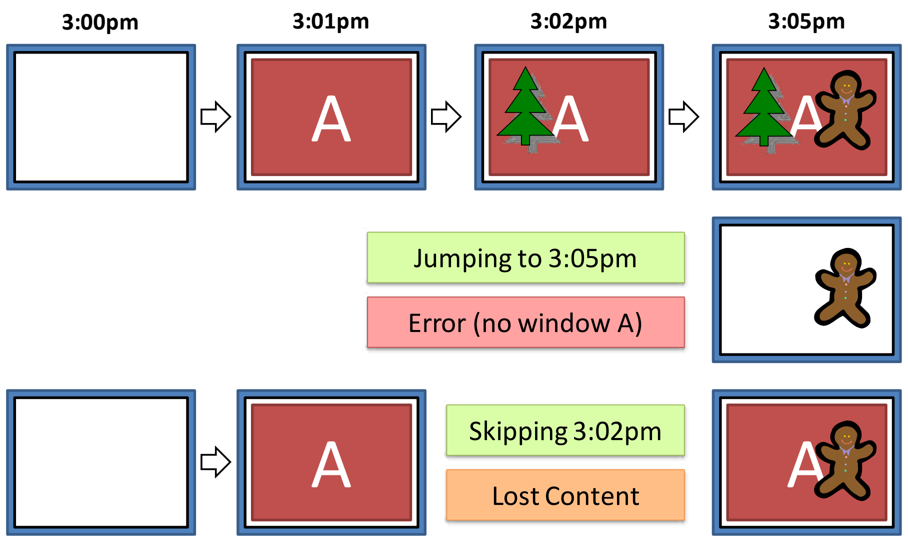


Figure 15 – Top: event sequence shown in full. Middle: starting playback of the log at 3:05 pm results in an error because the ginger bread man picture is being placed on a window that doesn't exist. Bottom: skipping over an event results in lost content, or content that isn't placed on the window that should have been.

To avoid problems caused by skipping events in the log, every event in the log must be replayed from the beginning. If a user is stepping backwards one step at a time, then the entire log file (up to the current position) must be reproduced at each step. Figure 16 illustrates some of the pitfalls in a display log playback.

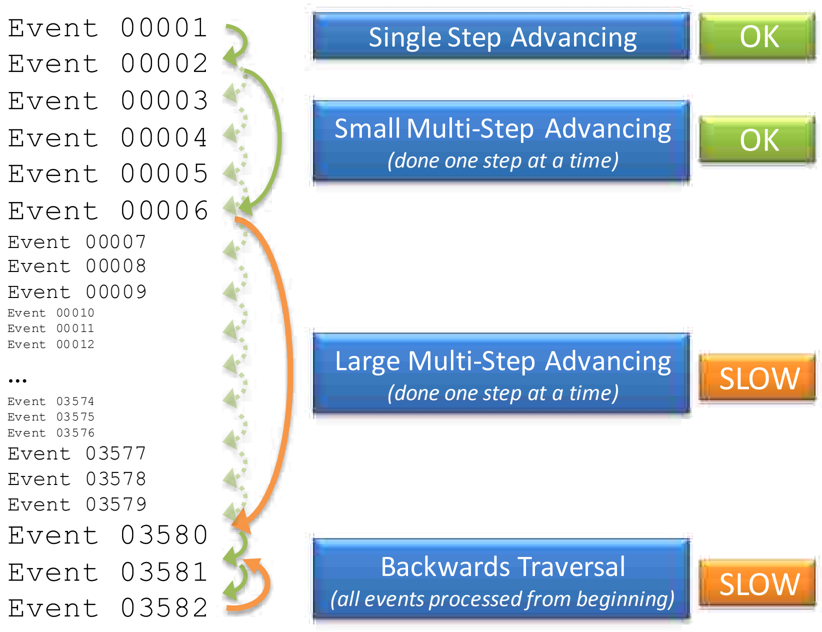


Figure 16 – A display event log requires every event to be processed in chronological order. This requirement can make traversal extremely slow in a couple of cases including: backwards traversal and large jumps in the log.

Key frames can be used to improve the speed of traversing a display log. A key frame is a list of all display events needed to reproduce everything exactly as it appears at a given point in time. Generating a key frame and marking its place in the log is equivalent to placing a shortcut entry point into the log from which no prior information is needed. During personal review/playback, key frames are only used as entry points when starting to view a capture, making large jumps in the capture, or traversing the capture backwards. For single step or relatively small advances in the capture, key frames are completely ignored.

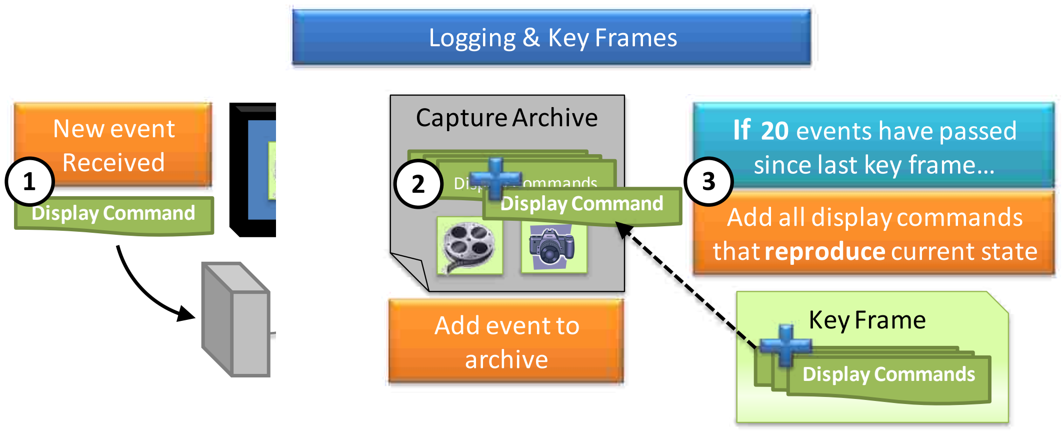


Figure 17 – Every time a display command is received, the display command is logged away as a display event. Then, the server checks to see whether it should insert another key frame – which is a set of display events that will reproduce the current display exactly.

Playable Media Tracking

A second difficulty with display logs comes when playable media is used during the meeting. Playable media refers to video and audio files that can be controlled with play, pause, and positioning. Each media's individual play state and position need to be tracked and derived separate from the review application's play state and position. Picking an arbitrary position in the capture will not automatically reveal whether the media is playing, paused, or 3 minutes and 21 seconds into the clip. Additionally, the usual playback strategy does not work for playable media. For example, a student may have the review mechanism paused but navigate to various points in the capture via his notes. Jumping to a place in the capture where a video was playing back would cause the video to start playing because there is an event in the display log indicating that the video started to play – this is incorrect behavior.

To find the current play state of a playable media element, the software must first locate the nearest key frame – this might be the beginning of the log. If the key frame contains a display event adding the media element to a window, then the initial media play state information can be read from the key frame. However, if the key frame does not include the media element of interest, then the media element will be added to a window at some point later in the log. When the media is added to a window, the play state is initialized to the default settings of the media (i.e. play=false; position=0.0; and play speed=1.0).

After the media state is initialized, all display events in the log, from key frame to seek position, must be considered. Each display event involving the media element must cause an update to the internally tracked media play state. Finally, the gap between the last relevant display event and the seek position in the capture must be evaluated. For example, in Figure 18 the review application is actively playing back the capture at position 4 seconds. The media has not been encountered yet. Jumping from step 1 to step 2, or 4 seconds to 18 seconds, a play event and a pause event are processed. The playback arrives at the new position instantaneously, but between the play and pause event, the media must be advanced by 2 seconds. Therefore, the derived media position is 2 seconds. In addition to counting all the time between known events, the gap between last event and seek position must be counted as seen in Figure 19. Step 2 shows that the gap is between second 20, where the media was set to “playing,” and second 25, where the reviewer is seeking to. Accordingly, the media’s position should be advanced 5 seconds (plus the 2 for the previous play span discussed for Figure 18).

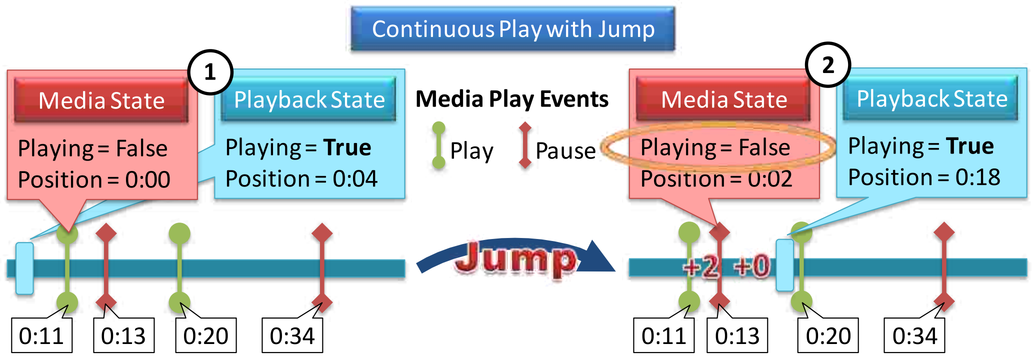


Figure 18 – Diagram of "jumping" or seeking while the review application is actively playing back. The derived media position is 2 seconds, based on 2 seconds of play and 5 seconds of pause.

On top of deriving basic properties of the media, the review application needs to reconcile the review application's play state with each media's individual play state. Default behavior would be to reproduce log events exactly as is. However, play state changes must be intercepted and changed based on the play state of the review application – if the review application is paused, then replaying a "play media" log event is not the correct action. For example, in Figure 19, the reviewer is sitting, paused, at 5 seconds when the user decides to "jump" or seek to 25 seconds. Jumping to second 25 in the capture means that the last event processed/known is the play event on the media at second 20. Normally, that event would be read out of the log and played back to the server without any alteration; however, because the reviewer is paused, the event is intercepted and changed to match the current reviewer state (paused).

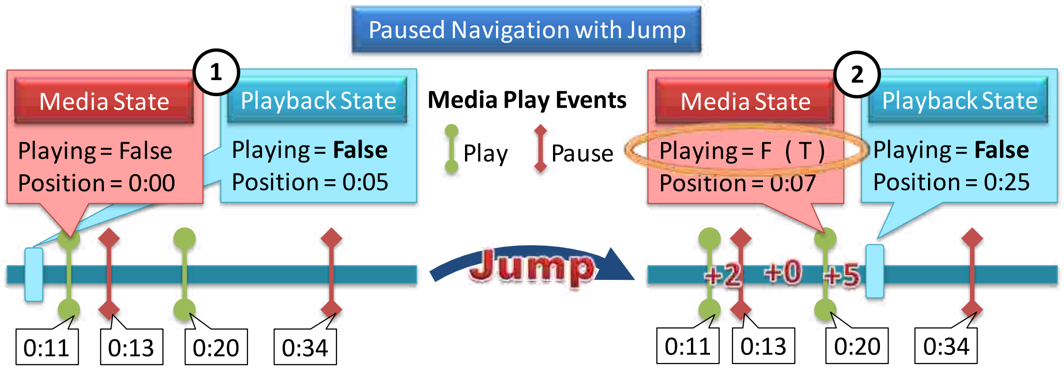


Figure 19 – Jumping positions in the capture while the reviewer application is paused causes the observable media play state to be **paused** while internally the media play state is **playing**.

Chapter 6 – Playback Controls

If the presentation capture was recorded like a movie, then the only play back option would be video playback. However, because a centralized display server receives all changes to the displays as separate, distinct events, many more controls can be offered. Knowledge about what changes occur where and when can allow users to more effectively browse the capture. Advancing by increments of change versus advancing by time increments is something to consider.

Playback Controls

Typical controls for reviewing a meeting capture are similar to video playback. One control possible is the ability to jump forward and backward by set intervals. For example, buttons can be provided that advance the playback position forward by 5 seconds, 30 seconds, 2 minutes, etc. Buttons like this give the user the power to skip over content rapidly. Although these time increments will get the reviewer roughly to where they want to be, it will not lock on to places of interest. Capture systems could facilitate locking on to interesting places in the capture if an extra post processing step has been taken. Without an extra step, the capture would have no concept of interesting versus uninteresting. For example, a system like Panopto [6], which does mark interesting places in the capture, must require either manual tagging or computerized analysis of the video to find pixel differences indicating a slide changes or some other similar method. This extra post processing can be assumed because the input to the capture system is a video stream which does not hold inherent awareness of slide changes.

In a system where changes to displayed content are received by the display device as individual events, interesting points in the capture correspond to those individual events. This method can be effective because changes to the displayed content during a meeting occur at a much more manageable rate than display changes occurring in filmed/animated movie. Software demonstrations cause the display to change more frequently than during a slide presentation, but do not usually demand movie-like refresh rates. Additionally, even if movies are used during the presentation, a logging based capture technique will ignore all the changes to the display and simply record when the video started and when it ended. Therefore, each display event recorded in the log can be used effectively to navigate the playback. Buttons that advance by 1 change, 5changes, or 20 changes can now be provided to the user both during and after the meeting because no post-processing effort is required to reference those interesting points.

Playback Timeline

The timeline is one widget that is fairly standard to playback controls. A timeline allows the user to grab the handle representing the position in the playback and move it around freely. A user can then jump to a specific time or position in the playback independent of where they currently are. For example, a youtube user can jump directly to his or her favorite scene in the video by clicking on the timeline at 2 minutes 34 seconds. Without prior knowledge gained through trial and error in locating that specific position, no user could identify the position 2m:34s as a point of interest.

With display events logged away in the capture, helpful indicators could be used to guide users in their timeline navigation. Robert Mertens [20] used this idea to mark areas of the timeline with varying blue intensities – the darker the blue the more potential interest of that region. In Mertens’s timeline (see Figure 20), footprints of how long users spent reviewing the capture over various regions are tracked with increasing blue intensity. Similarly, in this solution, for each display event, a semi-transparent tick is added to the timeline. Then, as events occur simultaneously or close in proximity, the darker the ticks appear to be. These ticks show where each change occurred and roughly how much of the screen space was changed (how many changes occurred at once).

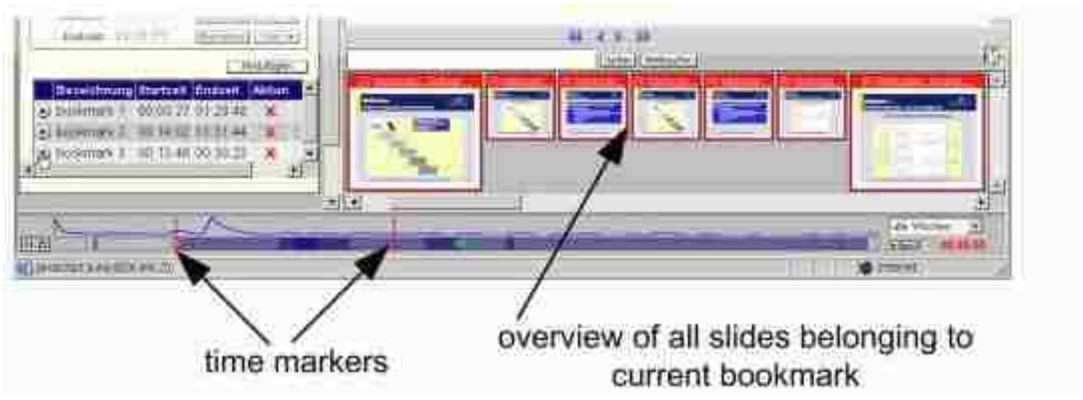


Figure 20 – Mertens's timeline with footprints indicating where various users have spent significant amounts of time reviewing.

Playback Speed

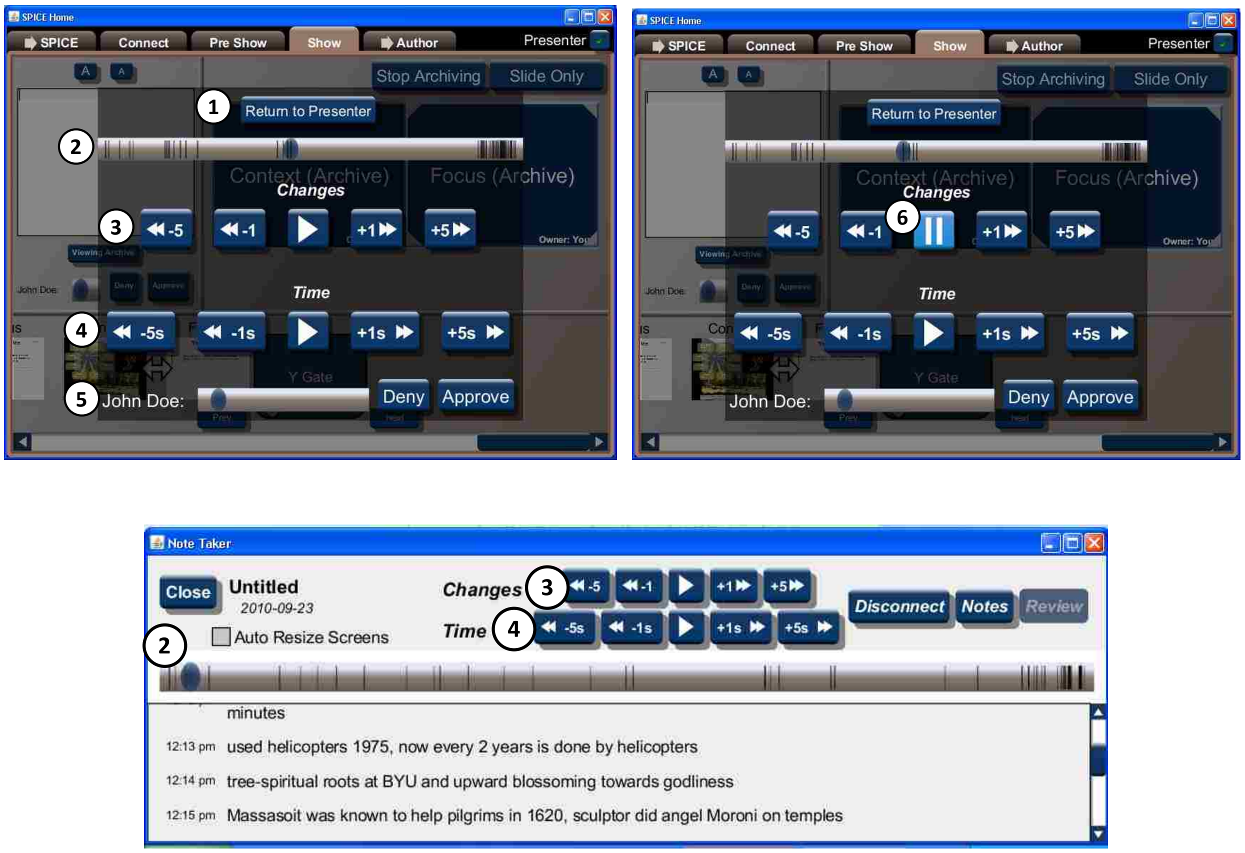
The expected behavior for media players and capture review applications is to play back the content in "real" time, or at the same pace at which it was recorded. However, just like a slow motion feature, or fast-forward feature can be useful, playing the capture back at higher speeds than real time can be useful. A few systems including the Tegrity [8] system facilitate adjustable rate playback. This allows users to scan over uninteresting sections of the capture to get to the information they need. Supporting this functionality is a straightforward process that is relatively common today.

Similar to adjustable rate playback, there is another way to scan a capture at a modified pace. During a meeting there can be anywhere from 10 seconds to 10 minutes of time passing between changes to the display. Consequently, playing back changes at a constant rate has practical value. Again, this is only possible when changes are clearly indicated in the capture – which is not the case in simple video capture. Playback by changes means that the capture position is advanced to the next change every second or pre-determined amount of time. Because each change in the capture has type information associated with it, the time spent waiting before advancing to the next change can be dynamically adjusted. For example, a software demonstration can advance every 0.1 seconds while a full image change can wait 0.75 seconds before advancing.

Implementation

Controls for advancing by both time and changes were built into the navigation system. The two sets of controls were grouped separately with a play/pause button of their own. One group was titled “Time” and the other “Changes”. This solution took care of the basic needs of viewing the capture while giving the user an enhanced set of tools to work with.

Figure 21 shows the controls provided to the user in both the Presenter role (top) and Note Taker/Reviewer role (bottom). The “Return to Presenter” button (1) hides the review windows, review controls, and reveals the normal presenter controls. The timeline (2) presents a clickable interface for jumping to any location in the meeting capture and shows where all display events occur via a black “tick” mark. The group of buttons under the “Changes” heading (3) allow for advancing +1, +5, -1, or -5 changes in the capture, and for playing back the capture by changes. The button controls in the “Time” grouping (4) play back the capture in real time and allow navigation by +1, +5, -1, or -5 seconds. As students make requests to rewind the meeting capture, a limited number of their requests will appear below the capture access controls (5) to show the name of the student and roughly where it will jump to. With each request, there is the option to approve or deny the request. If the capture is actively being played back, then the corresponding play button (6) will light up and become the pause button.



minutes
12:13 pm used helicopters 1975, now every 2 years is done by helicopters
12:14 pm tree-spiritual roots at BYU and upward blossoming towards godliness
12:15 pm Massasoit was known to help pilgrims in 1620, sculptor did angel Moroni on temples

Figure 21 – Capture access controls in both the Presenter and Note Taker/Reviewer roles.

There are additional controls for approving and denying rewind requests, and for viewing the archive without a rewind request that will be discussed in chapter 7. The controls described in this chapter are included in both the personal review software and the dynamic access controls provided to the presenter.

Chapter 7 – Live Review

In situations where participants are able to speak and convey all of the presentation material needed to start a discussion, rewinding the presentation (accessing the active meeting capture) may be unnecessary. Or if only very simple slides were used and can be navigated through quickly, then having the presentation capture on hand may not be needed. However, if either the professor is not incredibly quick at finding appropriate slides, or if a software demonstration was used during the presentation, then access to the active meeting capture has the potential to greatly aid the discussion.

Live Playback (during a meeting)

The desired interaction between presenter and participant can be seen by considering the example used in chapter 1. A student who hears about tRNA and thinks back on mRNA presented earlier in the lecture wants to know what pieces mRNA together. In the process of formulating his question, the student remembers the notes he took on mRNA, notes that say something like “DNA is unzipped, and then transcribed by mRNA” and “mRNA looks like half a strand of DNA when constructed.” The student guesses that there is where mRNA assembly occurs and therefore good context from which to ask his question. Selecting that note and clicking the option to request a rewind, he raises his hand. The teacher sees the hand and a notification on his presentation software indicating a rewind request. If the teacher is not ready to take the question, then he may ask that the question be saved until later, and continue on. When ready, the teacher can approve the rewind and call on the student for the question. All the material shown at the time the student took the note is then displayed for all to see.

As discussed in chapter 4, each note that a student takes is time stamped. When a request to rewind is made, the time stamps associated with the chosen note is adjusted to match the server’s local time before being sent to the server. In a simple system, this interaction would be all that is required to access the correct place in the capture. However, as described in the example above (student/teacher interaction), what is expected is that the presenter has the ability to direct whether or not a rewind occurs. Therefore, the presenter's software must provide the controls necessary for approving or rejecting such requests.

Rewind Request Management

Notifying the presenter of the rewind request is an important feature to include, but could put too much power into the hands of participants with malicious intent. Participants could easily send hundreds of rewind requests and flood the presenter with requests that may simply annoy or severely cripple the presenter's ability to present. A variety of methods were considered in order to guard against participants using this feature maliciously.

Social Solution

Ideally everyone in the meeting should behave according to the expectations of the social situation and environment. For example, in a university classroom setting, student are expected to be mature enough to not disrupt the professor as he or she teaches. However, this is not always the case. The social solution refers to resolving such disruptions face-to-face. This interaction is facilitated in that the name of the student is displayed for the presenter to see and use in singling out the disruptive student. However, this interaction fails when students do not answer to the name, real or made up, they provided to the system.

Auto Deny Option

A simple check box or toggle button could control whether incoming requests are received for approval or automatically denied. This would allow the presenter to put a stop to a barrage of requests from a malicious student. However, when this option is checked, it would also prevent others from making legitimate requests to rewind the capture. This technique could be adapted to block all requests from a specific user. This route has the potential to complicate the user interface; therefore, caution must be used if this option is implemented. The auto-deny option was not built in this solution except for automatically denying requests that are for the exact same position in the capture.

Select Decision Behavior: Immediate or Accumulate

If "Immediate" is selected, then requests will present themselves to the user to make a decision about the request immediately. If "Accumulate" is selected, then each request will accumulate in a list for later approval or denial. This option allows the presenter to not be distracted by requests when he or she wants to present uninterrupted. However, this idea must be combined with another one in order to actually prevent a large accumulation of erroneous requests from occurring. This idea was implemented with immediate request handling occurring by default, but allowing a simple click of the mouse/touch of the screen anywhere outside the decision buttons to cause the request to be ignored and accumulated.

Allow __ Number of Requests

With this option, the presenter can limit the number of request that will be presented to him – this is on top of an accumulation implementation. For example, the presenter may select "1" which would allow the first rewind request to be received and waited on while all other requests are automatically denied until the current 1 request has been arbitrated. Similarly all or any number of requests greater than 1 could be chosen as the limit. This idea was not implemented in this solution.

Implementation

Figure 22 shows the controls that were implemented in this solution. The request management controls appear over top of the presenter controls. This serves as an alert system for incoming requests. As part of the request management, there is the “Return to Presenter” button (1), the name and rewind position indicator (2), and the “approve” or “deny” buttons (3). Beneath the last request (4) is where future rewind requests will queue up. This queue is shown in both the request management screen as well as on a small side panel below the notes.

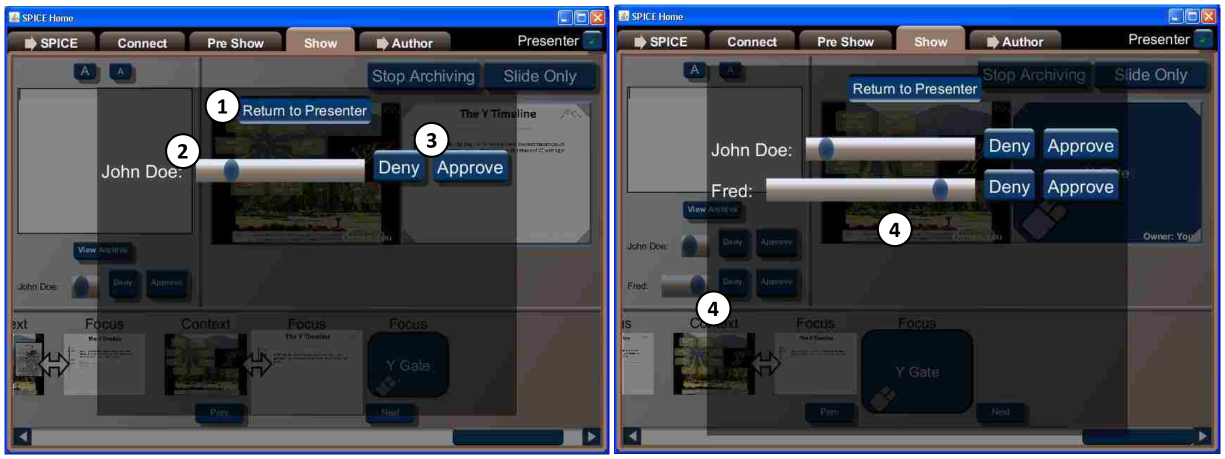


Figure 22 – The controls given to the presenter so that he can approve or deny each request to rewind.

The presenter also has the option to view the capture at any time he pleases – without needing a rewind request – via the “View Archive” button under the slide notes. Once the presenter approves a rewind request, the presenter application jumps into review mode and show the various capture access controls. The capture access controls are described in Chapter 6.

Chapter 8 – Personal Review

Personal review is a feature of this solution and a major consideration of this project because review is so important for retention and information retrieval. Goals unique to this solution include support for multiple screen presentations where reviewers have access to the media in an uncompromised form. In this chapter the architecture for the review application is laid out. The interface for viewing the capture in both detailed and contextual modes is presented in this chapter as well.

Review Application Architecture

As discussed in chapter 3, because all media passes through the display server and is saved in the exact form it was received, the media's original quality will be preserved. If a high resolution image with fine detail was shown on a large projector, then later, although it may be shrunk initially to fit a small screen, the detail in the high resolution image can be recovered. This is in contrast to a video capture technique where media is shrunk and compressed to fit a set resolution determined by the video encoding used.

The general architecture for reviewing a lecture capture is illustrated in Figure 23. The playback method is very straightforward and makes use of the same software used to host the presentation in the first place – the display server. As shown in Figure 23, the review application is in charge of launching a local display server (1), reading in log events and applying necessary transforms to them (2), and sending the events as display commands (3) – which are mostly unchanged – on to the display server. The display server does its usual function of receiving the display commands and rendering them to the display devices (4).

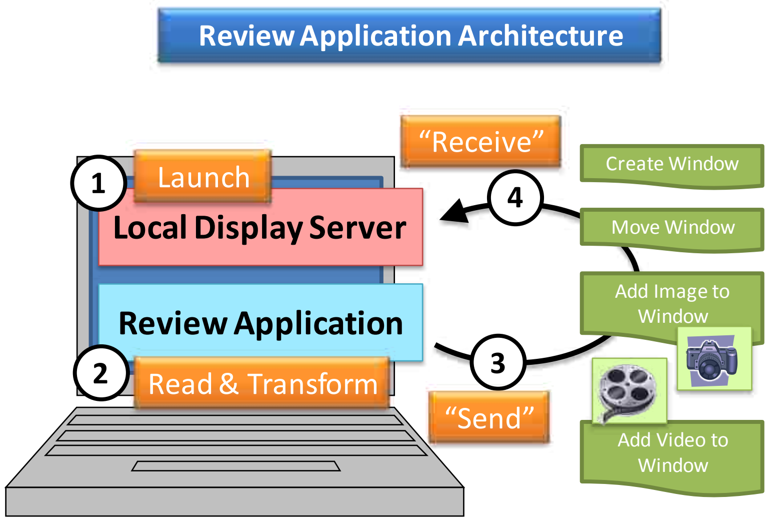


Figure 23 – The basic architecture for the review application.

Resize Functionality

In fulfilling the requirement that media quality must be preserved, it also follows that there must be a way to view the original resolution of the media. This can be done by "zooming" and "panning" on individual screens or the set of screens as a whole. To see the detail from a screen that is initially shrunk, the reviewer would have to 1) drag or click the zoom widget until sufficiently "zoomed," 2) click and drag the content itself or a view finder widget to the place of focus, 3) repeat from step 1 as needed. These controls can be slow and difficult for users unfamiliar with this interaction. A point and click method was chosen for simplicity's sake. The user moves the mouse over the screen they want to see better, receives feedback that the screen is clickable, and then a click will enlarge and shrink the screens accordingly.

Figure 24 shows conceptually the user interaction needed to resize individual screens. Without resizing any screens, all windows are evenly spaced over the width of the screen. After click #1, window B is expanded to fill in the available height. After click #2, window C is expanded and all other windows take up the remaining width of the screen. After click #3, window C is brought to full screen mode (the notes window is minimized), filling as much in both width and height as possible. Click #4 returns all screens back to their normal size.

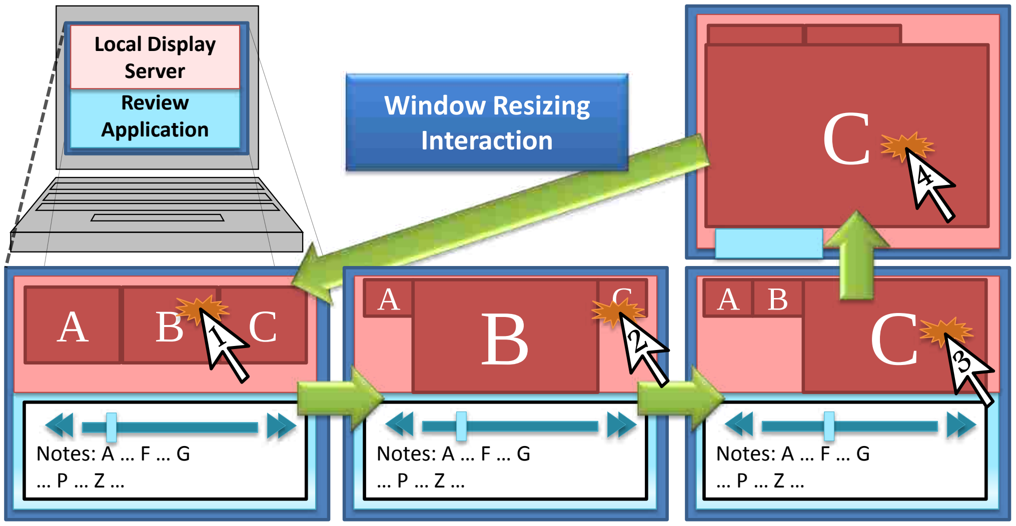
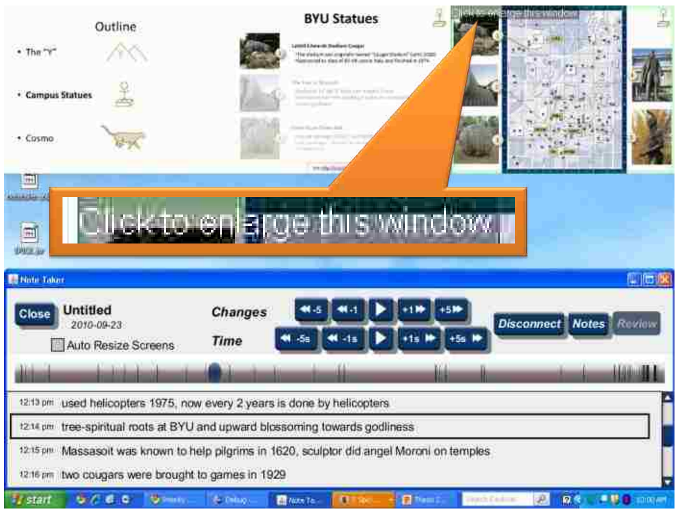
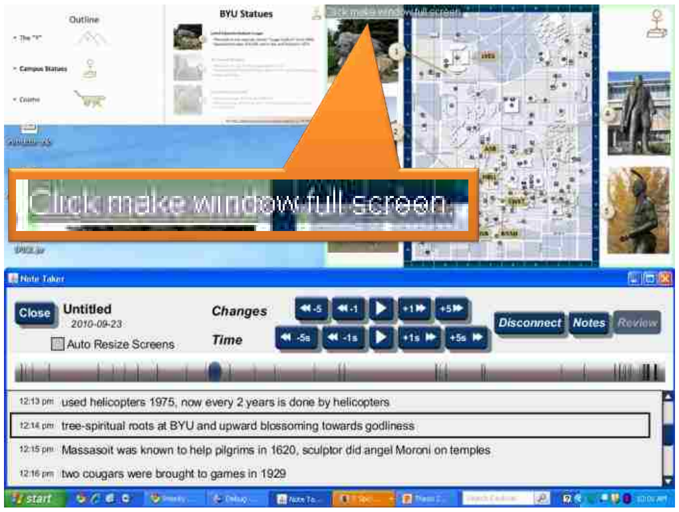


Figure 24 – Resizing interaction is a simple click selection of the desired screen to enlarge.

In Figure 25, there are a series of screen shots that show what the screen resizing looks like in practice. The feedback that indicates a screen is clickable is the green border that highlights the screen when hovered over. The text describing the action that clicking will trigger is also displayed. There are three actions that can be done to a screen: “Click to enlarge this window”, “Click to make window full screen”, and “Click to un-enlarge this window”.



12:13 pm used helicopters 1975, now every 2 years is done by helicopters
12:14 pm tree-spiritual roots at BYU and upward blossoming towards godliness
12:15 pm Massasoit was known to help pilgrims in 1620, sculptor did angel Moroni on temples
12:16 pm two cougars were brought to games in 1929



12:13 pm used helicopters 1975, now every 2 years is done by helicopters
12:14 pm tree-spiritual roots at BYU and upward blossoming towards godliness
12:15 pm Massasoit was known to help pilgrims in 1620, sculptor did angel Moroni on temples
12:16 pm two cougars were brought to games in 1929

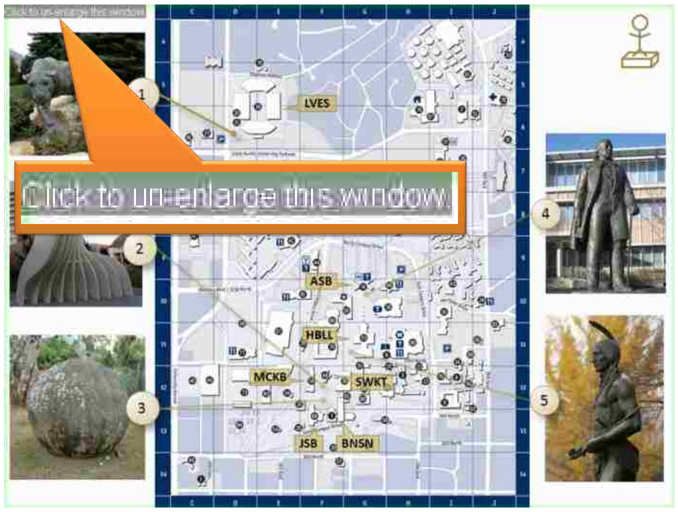
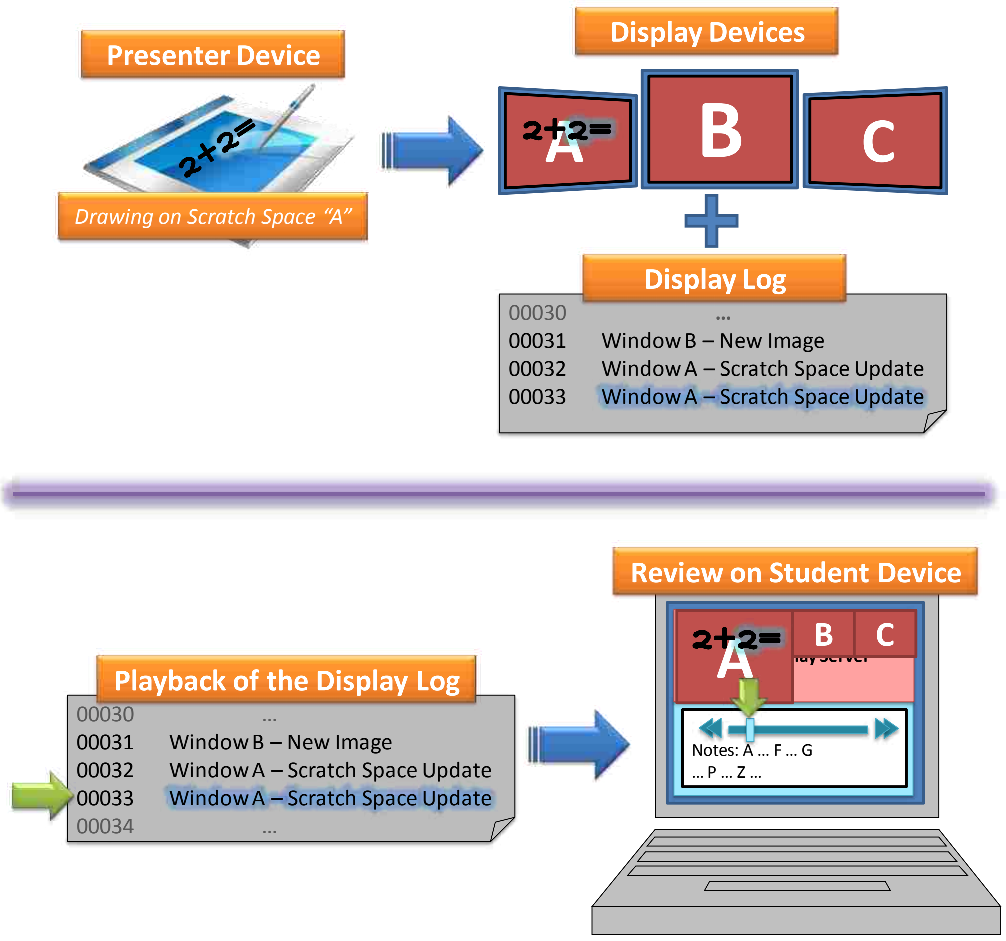


Figure 25 – Screen shots showing the various resizing prompts.

Automatic Resizing

While resizing can be done manually by the user, automated resizing is an easily added feature due to the nature of the meeting capture. During a presentation, the last screen or area of the screen that changed is likely to be the area of most interest to the viewers. Using this simple concept, the display log can be used to decide which screen should be enlarged.

For example, a math professor shows a slide stating and describing a theorem on the center screen (screen B in Figure 26). Next, using the left screen as scratch space for notes, the teacher begins writing out the proof. Each change to the scratch space is both rendered on the display device and logged away to the display log. When the slide was shown, the center screen was of primary interest. When the proof was written, the left screen became the focus. Later, during review, as the log dictates changes to each screen, it can also direct resizing based on where the last change occurred.



```
00030         ...
00031    Window B – New Image
00032    Window A – Scratch Space Update
00033    Window A – Scratch Space Update
```

```
00030         ...
00031    Window B – New Image
00032    Window A – Scratch Space Update
00033    Window A – Scratch Space Update
00034         ...
```

Figure 26 – Automatic screen resizing based on log events.

Resizing screens according to the logic described above will work for many but not all presentations. For example, a slide with text that changes to describe various pieces of a diagram that should remain the primary focus will cause the diagram to lose focus prematurely. For cases where the feature is not helpful, it can be turned off via the checkbox labeled "Auto Resize

Screens." Even with automatic resizing is on, manual resizing is allowed, but will be overridden any time a new change warrants an automatic resizing of the screens.

Chapter 9 – Evaluation

This chapter first addresses the formulation of the user study. Next, the demographics of the ten volunteers are summarized. The user study's agenda is then laid out with a description of the various tasks that each user completed. Finally, the results and feedback show that users were successful in engaging in the rewind interaction and finding detail from the meeting capture.

User Study Formulation

Using the active presentation capture during a meeting for review has not been accomplished by any published system at the time of this paper's authoring. Through actual implementation and use of the solution we hoped to verify that the ideas behind dynamic capture access and multiple screen review are viable. Therefore, the solution was implemented and a task based user study was formulated. The purpose of the user study was to answer the following two questions:

1. Can audience members use notes that they take to initiate a rewind of the presentation?
2. Can audience members use the note taking software to review a multiple screen presentation on a significantly smaller device in order to retrieve detailed information from the presentation capture?

Participant Demographics

To identify whether this solution actually facilitates people engaging in this unique interaction and enhanced multiple screen review, ten volunteer users were brought in individually to complete certain tasks. Each person was solicited through word of mouth, an email or a Facebook general invitation. Although male and female, student and non-student were solicited 9 of 10 were male and all were students. Four of the volunteers were working on graduate degrees; areas of study ranged from health and wellness to accounting to law to fresh water ecology to computer science – 8 people with unique areas of study while 2 studied business. While not extremely diverse in age or occupation (students), having an all student population fits well with the group of users this solution would be targeting.

Each participant came with the expectation of helping a graduate student out and receiving an unspecified treat as a thank you (the treat was a couple of candy bars of their choosing). No monetary compensation was given. The two questions this study intended to answer were not revealed to the participants. During solicitation and training, participants were told that they were to help simulate a classroom experience using a note taking system and that their interaction with the software would be observed.

User Study Agenda

Each study session was kept to 30 minutes or less with the following agenda:

Table 1 – User Study Agenda contains the outline for the script used during each user study session

User Study Agenda
• Agenda Overview • Training on ...   ○ Note taking   ○ Right clicking a note to request a rewind   ○ Using the reviewer controls to access a presentation capture • Tasks explanation for…   ○ Ask a question during the presentation   ○ Answer 3 questions after the presentation • Presentation on “BYU Fun Facts” (with simulated “student” questions) • Review/Student takes a 3 question “quiz”

- Feedback

Training was simple and brief, but was a hands-on experience including a short and simple presentation in which they took at least one note and clicked on “rewind.” Each person was told that we wanted them to simulate a student taking notes and asking a pre-meditated or formulated question during the presentation based on one of their notes, and that the question would be handed to them on a sheet of paper at the appropriate time and place. These simulated questions (two of them) made up the first half of their tasks. The second half was to answer a review questions after the “lecture” using their notes and presentation capture. Training and task explanation together took less than 10 minutes.

The subjects were given a presentation on fun facts about Brigham Young University. Media used in the presentation included: bullet points and other text, various images and generated graphics, a short movie of Cosmo’s stunts (BYU’s mascot), and internet browsing showing a statue through Google Maps and yfacts.byu.edu – from where most of the presentation was derived. All of this media was available for review both during and after the meeting.

To avoid having to solve an unrelated problem of “how do we distribute the meeting capture to a large number of attendees?” a thumb drive was used to transport the capture to the subject’s machine. Once on the machine, the reviewer application locates the correct capture to use based on the information gained and stored from the display server during note taking (see Chapter 4). From there each subject used the reviewer application to answer 3 simple questions.

Results and Feedback

In addition to feedback questions about their user experiences, subjects were video-taped so that later review could be done on their actual usage. Feedback questions were aimed at gaining the thought processes motivating the interaction or lack of interaction they had when using the software. Most users were able to request a rewind without any problems and with little help. By the second simulated question during the presentation, with the exception of one, all users were able to make the rewind request without any help at all. One volunteer of his own initiative asked two additional questions using a rewind request each. Two others asked an additional question also using the rewind request feature.

Table 2 – Simulated question asking with rewind request - Per question, there were 10 users who successfully requested a rewind when presented with the question they were to ask. The amount of help each one needed varied from re-training to a simple prompt of "Do you have a note?" to no help at all.

Question	Needed Help	Needed Prompt	No Help, No Prompt
#1	3	3	4
#2	1	0	9

There were unexpected results in the review section of the user study. Only 3 out of the 10 participants used the manual re-sizing feature in order to view content from the presentation. One person needed to resize a screen in order to see the statue map's detail, while another need it to read the slide with the cougar mascot's nicknames and the other person needed it for both. The rest used other techniques for answering the questions such as leaning in close to the screen, reading their own notes, relying on their memory, and listening to the audio. At least 5 participants seemed to rely on the audio in order to answer some of the review questions. The majority of the users did have the "auto-resize" feature on while reviewing, but, according to feedback responses, did not consciously turn it on or recognize that it was on. This seems to indicate that auto-resizing work naturally enough for most users and, at least for the test presentation, was sufficient resizing for the reviewers.

The following bullet points summarize the feedback for improvements and suggestions provided by the participants. Each point of improvement would be great points for future work to address.

- Notes being completely separate was confusing (pressing enter didn't jump to a new line, but rather a new text field)
- The option for requesting a rewind should be more obvious and named something like "Ask a question"
    - Allowing the student to type the question or topic for discussion might be useful
- Playback controls were a bit overwhelming (too many options)
- Re-sizing features needed to be more obvious

Positive feedback for the user study goals concerning the rewind request feature and linked notes surfaced in many of the participant's comments. Four users noted in particular that the rewind feature would help both students and professor to be on the same page with questions and discussion, two of them citing actual experience where confusion could have been avoided had this system been used. One participant who took very detailed notes remarked that this would change the way he took notes – there would be no need for verbose notes, but rather simple indexes into the presentation capture.

Conclusion

Participants successfully engaged in the dynamic capture access with extremely limited training. Three participants went beyond what was asked of them and requested rewinds because they wanted to find out more BYU trivia. These results clearly indicate an affirmative answer to the question "Can audience members use notes that they take to initiate a rewind of the presentation?"

The answer to the second question concerning capture review on a smaller display device appears to be affirmative as well, but is less clear-cut. Participants, without question, *were* able to retrieve detailed information from the capture; however, most did not need to rely on great image quality from the presentation – answers were found through the audio or notes captured during the meeting. The fact that three users did rely on the detail from a few of the images does seem to suggest good reason to preserve and provide access to as much detail as possible.

Both questions that the user study intended to answer were answered affirmatively. The observations and feedback gained from the study provide useful ideas and targets for future work. Areas of meeting capture involving dynamic capture access and multiple screen review are relatively unexplored; therefore, this solution and lessons learned from this project could be a useful starting point for future development.

Chapter 10 – Conclusion

The solution presented in this paper addresses two major issues found in meeting capture by centralizing the display work done during the meeting and introducing dynamic access of the active capture. The first issue is the general lack of support for multiple-screen, multiple-media presentation capture and review. The second issue is capture systems do not facilitate any support for asking question about and discussing material presented during the meeting except in certain limited circumstances. The ideas and solution presented in this paper are intended to show the feasibility of dynamic capture access, demonstrate one solution for that problem, and provide various intermediary solutions needed to support both live access and review access of a multiple screen presentation. Participants in a user study also gave valuable feedback for our solution.

Nine out of ten volunteers were able to complete the rewind request without any help on their second attempt. Three others asked additional questions using the rewind feature facilitated by the system. The study performed here does not indicate an improvement in classroom learning. However, it does suggest that a meeting capture solution can be built such that an average student can initiate and direct dynamic capture access during a meeting. Future work could be done to study whether or not learning is enhanced by such a feature. Other work in this area might include a change from “requesting a rewind” to “asking a question” and other options that can be explored from that.

The display log and simple screen resizing did allow for sufficient detail to be deciphered from the capture, but was not isolated in the user study well enough to show ease of use or overwhelming need. As discussed in Chapter 2, the only system that allows flexibility in the number of screens (3+) and viewing size of individual screens was Qumu. While Qumu may often provide *sufficient* detail, there are cases where it will not. The solution presented in this paper is different – if there was sufficient detail during the meeting, the media can be enlarged to see equivalent detail. Future work could be done to look at making the resizing features more apparent. Other work could be done to investigate the potential for a purely video based solution.

Bibliography

1. *DimDim*, dimdim.com, 2007.

2. *Echo360*, echo360.com, 2010.

3. *GoToMeeting*, Citrix Online, 1997-2010.

4. “High-definition video,” en.wikipedia.org/wiki/High-definition_video, November 2010.

5. *Matterhorn Project*, Opencast, 2010.

6. *Panopto*, panopto.com, 2010.

7. *Qumu*, qumu.com, 2010.

8. *Tegrity*, tegrity.com, 2010.

9. *TideBreak*, tidebreak.com, 2004-2010.

10. *WebEx*, Cisco, 1997-2010.

11. Anderson, R., Chung, O., Davis, K.M., Davis, P., Prince, C., Razmov, V., and Simon, B., “Classroom Presenter—A Classroom Interaction System for Active and Collaborative Learning,” In *Proceedings of the Workshop Impact of Pen-Based Technology on Education*, 2006.

12. Anderson, R., Simon, B., Wolfman, S., VanDeGrift, T., and Yasuhara, K., “Experiences With a Tablet PC Based Lecture Presentation System in Computer Science Courses,” In *Proceedings of the ACM Special Interest Group on Computer Science Education*, 2004, pp. 56-60.

13. Abowd, G. D., Brotherton, J., and Bhalodia, J., “Automated Capture, Integration, and Visualization of Multiple Media Streams,” In *Proceedings of the 1998 IEEE Conference on Multimedia and Computing Systems*, 1998, pp. 54–63.

14. Chiu, P., Kapuskar, A., Reitmeier, S., and Wilcox, L., "Notelook: Taking Notes in Meetings with Digital Video and Ink," In *Proceedings of ACM Multimedia*, 1999, pp. 149-158.

15. Chiu, P., Liu, Q., Boreczky, J., Foote, J., Fuse, T., Kimber, D., Lertsihichai, S., and Liao, C., "Manipulating and Annotating Slides in a Multi-Display Environment," In *Proceedings of InterACT*, 2003, pp. 583-590.

16. Demeure, I., Faure, C., Lecolinet, E., Moissinac, J.C., and Pook, S., "Mobile Computing to Facilitate Interaction in Lectures and Meetings," In *Proceedings of the First International Conference on Distributed Frameworks for Multimedia Applications*, 2005, pp. 359-366.

17. Denoue, L., Hilbert, D., Adcock, J., Billsus, D., and Cooper, M., "ProjectorBox: Seamless Presentation Capture for Classrooms," In *Proceedings of World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education*, 2005.

18. Kam, M., Wang, J., Iles, A., Tse, E., Chiu, J., Glaser, D., Tarshish, O., and Canny, J., "Livenotes: A System for Cooperative and Augmented Note-Taking in Lectures," In *Proceedings of the CHI: ACM Conference on Human Factors in Computing Systems*, 2005, pp. 531–540.

19. Lienhard, J., and Lauer, T., "Multi-Layer Recording as a New Concept of Combining Lecture Recording and Students' Handwritten Notes," In *Proceedings of ACM Multimedia*, 2002, pp. 338-342.

20. Mertens, R., Brusilovsky, P., Ishchenko, S., and Vornberger, O., "Time and Structure Based Navigation in Web Lectures: Bridging a Dual Media Gap," In *Proceedings of the World Conference on E-Learning in Corporation, Government Health, and Higher Education*, 2006, pp. 2929–2936.

21. Richter, H., Geyer, W., Fuchs, L., Daijavad, S., and Poltrok, S., "Integrating Meeting Capture Within a Collaborative Team Environment," In *Proceedings of the International Conference on Ubiquitous Computing*, 2001, pp. 123–138.