Chapter 7

Client Applets

A recorded E-Chalk lecture is replayed by Applets. When the remote user opens the lecture's Web site, one Applet per transmitted data stream is opened: a board Applet, an audio Applet and a video Applet, or any subset of these three if not all of the streams were recorded. The same Applets are used for live transmission and replay of archived lectures, only the Applet parameters given in the Web page differ. If the transmission is live, each client Applet opens a TCP socket connections to the E-Chalk server and read the data from this connection to display the current lecture data.[1] For replay from an archive, an additional Applet is opened. The control-panel Applet provides VCR operations on the recording, see Figure 7.3. Timed slide shows can also be combined with lecture recordings, see Section 6.13. Displaying them is handled by the slide-show Applet.

For live transmission, the client Applets read the lecture data through a socket connection from the E-Chalk server program. Because Java Applets can only connect to the server their class code is loaded from, the E-Chalk server must run on the Web server, or the Web server must act as proxy for E-Chalk's server sockets. For an overview to the client-server connections for live transmittions, see Figure 7.1.

When the lecture is replayed from archive, all data are loaded using the Web server's HTTP service. As described below, the board, video, and slideshower Applets can synchronize their replay with the Audio stream. The reason for adapting everything according to the audio stream is that interruptions in the audio stream are perceived as quite disturbing compared to stalls in the video or board replay. See Figure 7.2 for an overview.

While the standard setup is to access a lecture with a Java-enabled browser or with the Java `appletviewer` by an HTTP address, the client Applets can also be run from the local file system, for both live and archived lectures. Of course, for the live session the E-Chalk server must then be running on the same server as the client.

[1] An early implementation of E-Chalk also allowed a live connected shifted in time for late connects. When remote viewers connected ten minutes after the start of the recording, they had option to get the lecture recording presented as a replay, shifted ten minutes in time compared to the live stream. Since this feature was never used by real users, it is no longer supported in E-Chalk .

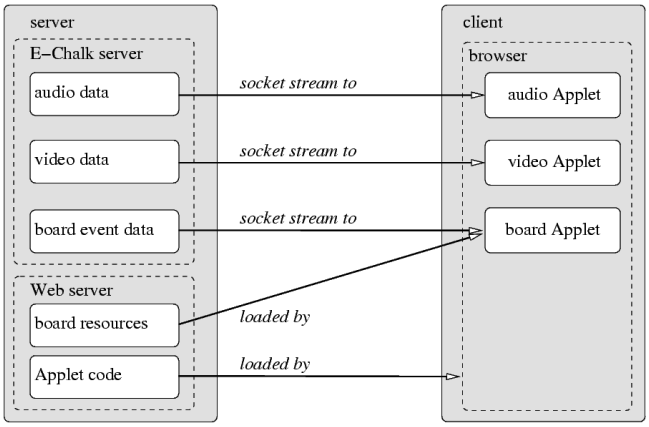

Figure 7.1: Communication between an E-Chalk server and a client for live transmissions. Board resources include images and data of Applets added to the board.

7.1 Client Control Panel and Masi Interface

To be able to synchronize the different replay Applets, the abstract Applet subclass `de.echalk.applet.Masi` (Media-Applet Synchronization Interface) was defined and all the content displaying Applets inherit from this class. A `Masi` Applet provides methods which report the capabilities for navigation in time (like "can jump forward", "can jump backward", "can pause") and the current replay status (like "is paused" and the current offset in time). Also, it provides methods for modifying the play status (like modifying the play offset or toggle pause mode).

The control panel scans for all `Masi` Applets started in the same Web page and allows the user to access any controls that are simultaneously provided by all `Masi` instances.[2] In the E-Chalk client Applets, all capabilities for navigation in time are supported: pausing, setting forward and backward offsets, rewind, and jumping to random offsets.[3]

The control panel displays the time reported by the `Masi` Applets (if the Applets' local times differ, the control panel uses the maximum time value) with both a slider and a text display. Clicking on the time display toggles the display modes of elapsed time, remaining time, and total time. The control panel displays a warning text at the bottom and changes the mouse cursor to a wait cursor whenever the `Masi` instances are stalled, for example when they do not get their data fast enough on a forwarding action.

When a `Masi` instance reports permanent errors (by its method `boolean hasPermanentErrors()`) or if it is no longer active (determined by the `Applet` method `boolean isActive()`), it is removed by the control panel from the list of controlled Applets. When no Applet remains to be controlled, the control panel closes.

[2] Except for navigation by *chapters*, which is defined in `Masi` but currently supported neither by the control panel nor by the replay Applets.

[3] A possible improvement would be a capability to play at higher speeds.

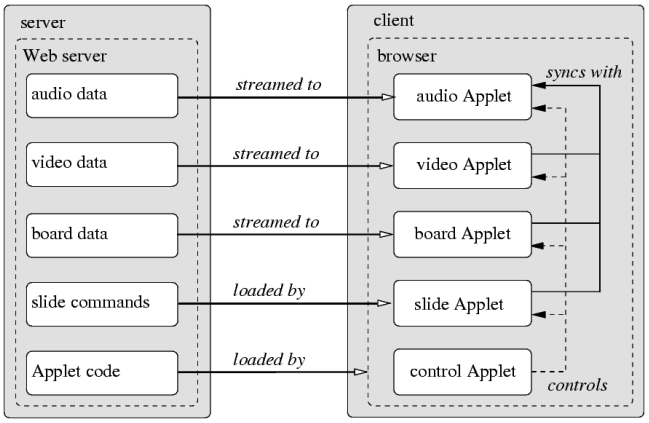

Figure 7.2: Overview of the client Applets for replay of recorded lectures.

The Applet parameters recognized by the client control panel are:

- `seconds`

  This is a mandatory parameter, defining the length of the recording in seconds as a decimal integer.

  The information is given to the control panel as parameter instead of reporting such a information with `Masi` method, as E-Chalk data streams do not report their total length at the beginning of the data.[4]

- `title`

  This is an optional parameter defining the lecture title to be displayed.

- `initfile`

  The optional init file (with the given path relative to the control panels code base) defines a skin for the control panel. The init file is a Java property file with property entries defining the position and sizes of the control panel's control elements as well as the positions and colors of text font (including a "shadow" color), given separately for the title text, the time display, and for warning messages printed at the bottom. Also, an image file which contains the control panel's background image with inactive buttons, and the images of the control elements buttons for pressed and for mouse-over look is contained. For the play/pause button, appearance for both play and pause modes are defined. It may also contain an image for filling the slider track. See Figure 7.4 for example skin graphics.

- `autoplay`

  This is an optional flag to determine if the `Masi` instances should start immediately by the control panel when their initialization phase is finished. Otherwise the control panel will set the replay to pause mode at start and it must be started manually by the user. The parameter defaults to true.

[4] In fact, this is not possible with streaming formats.

Figure 7.3: A snapshot of the client control panel frame with mouse-over effect for the play/pause button.

- `x` and `y`

  These are optional x and y-coordinates as decimal integer for the top-left position of the control panels frame. By default, the position will be determined by the users window manager.

7.2 Board Client

7.2.1 Event Handling

As described in Chapter 4, the board client uses the same classes as the board server component, just without the authoring elements. It uses the `DrawPanel` component[5] to interpret the events. In addition to several classes the board client shares with the server side board, the client board uses four classes.

First, the Applet class `echalk.client.Client`, which is the main class of the client. It inherits from `Masi` (see above), realizing the interfaces for control by the control panel, handles the Applet's parameter-specific setup, and controls the replay. An instance of the inner thread class `Client.EventReaderThread` concurrently reads all the events: for live transmissions, that is done from a server socket[6], for a recorded lecture, directly from the event file[7]. The event-reading thread submits the threads to the `echalk.client.EventScheduler`, which handles the timed delivery of events to the `DrawPanel`. For a recorded lecture including an audio stream, a thread `EventScheduler.AudioSyncThread` is started, adjusting the local board time at regular intervals (every five seconds) to match the audio client's time and thus preventing a slowly accumulating synchronization offset.

7.2.2 VCR Operations

While the audio and video stream need only the stream data of the current point of time t_0, the event-based nature of the board means the board needs to examine all events in the interval $[0, t_0]$ to construct the board view for t_0. When the board has to jump from a time offset t_0 to an offset $t_0 + \Delta t$ in the future,

[5] See Section 4.1.
[6] See Section 4.11.
[7] See Section 4.10.

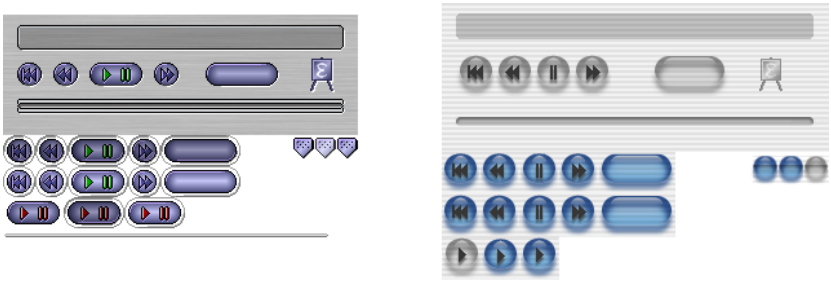

Figure 7.4: Two example skin graphics, left the default skin and right an OS X Aqua–style one. Transparent image parts are shown in white.

the scheduler has to try to submit all missing events (events with timestamps in $[t_0, t_0 + \Delta t]$) immediately. This causes the board to stall if the event reader has not yet read the events up to $t_0 + \Delta t$. Fortunately, this is usually only the case if a replay is forwarded a considerable amount of time at the very beginning of replay, and even then only for a short time, because the board event data are relatively small in size.

Because there is no general reverse mechanism for a board event, jumping back to a offset of $t_0 - \Delta t$ is realized as a total rewind (jumping to offset zero by clearing the whole board) and then jumping forward to $t_0 - \Delta t$.

To speed up jumping to a new offset, the changes in the board content are not shown immediately. Instead, the changes are applied to the offscreen buffer. Intermediate changes are only shown by repaints during lengthy VCR operations (when the adjustment takes more than one second to realize), and only in steps of one second, giving the user some feedback that the board is still active. The board also signals the user to be busy during the adjustment by changing the mouse pointer to a wait pointer.

7.2.3 Scrolling

The server board can be scrolled with mouse drags by the *drag handles*[8], while other mouse-drag actions are used to draw on the board. The client board, which the remote viewer cannot paint on, allows to do scrolling-type drags everywhere on the board.[9] To give a feedback to the user, the mouse cursor changes to move pointer defined by the operating system once he or she starts to drag. Horizontal drags are also possible for client boards with a horizontal extension smaller than those of the server. This can happen when the client screen has a smaller resolution or because the remote user resized the board.

For vertical scrolls, the client has two possible sources, the user drags or the transmitted scrolls from the lecturer at the server board. By default, the client combines the two, always using the last defined scroll offset, regardless of it source. This implementation assumes the standard case of the remote viewer looking at the board section the teacher uses for the given lecture portion, while still allowing them to peek at older portions.

[8] See Section 2.4.

[9] The only exception are the areas of embedded Applets, because the Applets consume those mouse drags themselves.

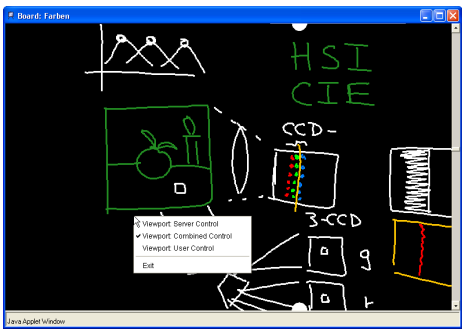

Figure 7.5: Board client with context menu.

A pop-up menu in the client allows the user to change the mode of handling scrolls. The behavior may be changed to handle only user drags or only transmitted scrolls, see Figure 7.5.

7.2.4 Handling Applets

The Applets are managed basically in the same way as described in Section 4.9 apart from recording facilities, which are obviously not available at the client side.

The local user can interact with the Applet regularly. While this can be used as a feature for adding interactive elements to a lecture, this is a major problem one wants a close reproduction of the live lecture, as the remote users interactions will usually not fit together with the lecturer's recorded Applet interaction. See Section 4.9.1 for a discussion of the Applet replay problems.

For the Applet class to be loadable on the client side, they must located in the board client Applets `classpath`. For replay from a repository, this can be done by putting the Applet files into one Jar archive with the code for the E-Chalk clients. For live transmissions, this is not possible, because the Jar archive is loaded once when the remote user connects, and the Applets to be used in the lecture are not yet stored by the server. In this case, the `codebase` parameter should be used to load all the classes, both E-Chalk client Applets and board-integrated Applets, from the `applet` directory. Therefore, the E-Chalk code must then reside there, too.[10]

7.2.5 Board Parameters

There are no mandatory Applet parameters of the board client Applet, all are optional. The parameters are:

- `port`

  Giving this parameter marks a live transmission. The parameter is the decimal port number to connect to on the live server. The server must

[10] For the current implementation of the E-Chalk system, this is not automatically handled. The template handling in Section 3.7 can be easily configured to handle the live case, but bundling together the Applet classes in a Jar archive is not yet supported and has to be done manually.

be the Web server the Applet is loaded from since Java's security model does not allow unsigned Applets to connect to other servers. If Web server and E-Chalk server are not running on the same machine, the Web server must be running a proxy to forward the E-Chalk live streams between E-Chalk server and E-Chalk client Applet. Otherwise, live transmission is not possible. Of course, this restriction also holds for live transmissions without the board stream, for example if only the audio signal is streamed.

The `port` parameter has no default value, meaning that replay from archive is assumed as a default. If the parameter is set, but a parse error occurs, or if the port number is out of the valid range, the parameter is set to 9996, the server's default port number.

- `delayoffset`

  This parameter is used to give a delay in milliseconds for displaying live events; it is ignored if the board does not run in live mode. The delay is introduced to compensate for the delay the audio client needs for buffering. The default value is 12,700 ms, which is the live delay used by the old WWR2 implementation of the audio client. See Section 7.3 for details.

- `masiclient`

  The Applet name given to a `Masi` instance to synchronize with, usually the audio client Applet.

  If a name is given and if the board client can find a `Masi`-extending Applet with the given name in its `AppletContext`[11], it starts a thread that ensures synchronization with the named Applet as described in Section 7.2.1. The parameter has no default, meaning that no audio stream is assumed to exists if the parameter is not set.

- `topleftx` and `toplefty`

  These are optional entries for explicitly positioning the client-board's window on the screen by defining the values of the window's top left corner. If not given, these parameters default to zero.

- `scrollbar`

  This is a flag causing a vertical scrollbar to be added to the board window. This provides a standard GUI element for vertical scrolling in addition to the drag feature. The scrollbar slider's position and size give a visual feedback of the board area's position in the total board history. On the other hand, because the scrollbar needs a certain amount of space, this can make the vertical space available on the user screen too small for the board.

  If not set, this parameter defaults to `true`. In the standard HTML templates of the E-Chalk system, the flag is set to the same value as the scrollbar flag in the server, giving the remote user a scrollbar exactly when the lecturer used one.

[11] See Section 4.9 for details on `java.applet.AppletContext`.

- `autostart`

  When this flag is set to `false`, the board client starts in pause mode and waits to be externally triggered via with the `Masi unpause` method for starting to play. If the control panel Applet is used, it expects the client Applets to wait to be started (and therefore letting the control panel synchronize the start time), if no control panel is present, the client will have to start automatically or it will pause indeterminately.

  The default value for this entry is `true` for live transmissions (when no control panel is present) and `false` for archived replay (when the control panel Applet is used for control).

- `embedded`

  Setting this parameter to `true` causes the client board to use the Applet area in the HTML page rather than opening its own frame. The default value is `false`.

  This parameter was introduced due to user requests. In a distance teaching project at Universität Regensburg, users wanted to combine video and audio streams in Real format [79] with a timed display of Web pages (triggered with SMIL [W3C98, W3C01]). Some of the pages were to contain short board recordings. With their remote viewer already operating both a *RealPlayer* window and a Web browser, they wanted to avoid another window for the user to operate.

- `autoclose`

  A client is notified of the end of a live transmission by a `terminate` event, see Section 4.10.2. By default, or when `autoclose` is set to `false`, it opens a message dialog to notify the user. This is especially important for transmissions without audio signal, as the remote viewer would have difficulty in distinguishing between a transmission that has ended, one that stalls, or one where the teacher simply pauses for a while. Note that for replayed lectures this is usually not needed due to the presence of the control panel Applet, showing the lecture's position in time.

  When the `autoclose` parameter is set to `true` for a live transmission, the board frame closes when the lecture ends. For the default setting, this is not very useful, as keeping the board open allows the student to browse through the board drawing. In fact, this parameter was introduced for a special setup where the board client is opened from the server side, see partt on setup of FU data wall in Section 8.1.1.

- `yoffset`

  This integer parameter causes the client to shift the vertical scroll-offset by the `yoffset` value. The default value is zero. The parameter is only used in a special setup where the board content is displayed with multiple board Applets. See part on setup of FU data wall in Section 8.1.1 for the usage example.

- `server`

  In practice, this parameter is only used for debugging purposes. It gives an IP address or host name to connect to in live mode. Note that an

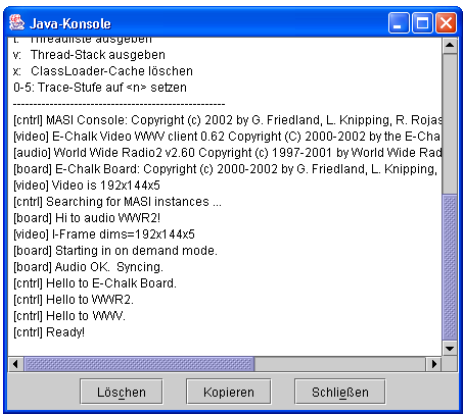

Figure 7.6: Standard client messages for debugging purposes written to the Java console of the browser.

Applet is usually only allowed to connect to the host its code is loaded from. As a consequence, this parameter is only used when starting the client from a local HTML file. For this the Java `appletviewer` and some browsers allow to loosen the security restrictions to connect to the local host.

When the Applet is loaded from a remote connection, the server location of the Applet's code-base entry is used instead. When the Applet is loaded locally, the server parameter is used instead, or if it is not set, its default value of `127.0.0.1` is used.[12]

- `baseurl`

  This parameter is used as the root location for all board data to be loaded: the events data file `board/events`, the image data files `images/`*no*`.dat`, and the Applet HTML files `applets/`*no*`.html`. It is given as a relative path to the code-base URL of the Applet and defaults to the code base. Because of the security restrictions of Applets, it has to be the code base itself or a subdirectory.

  The parameter is used for debugging purposes only.

- `debug`

  The board client writes certain status messages to the browser's Java text console. See Figure 7.6 for typical messages produced by the client control panel and the three E-Chalk clients, board, audio, and video.

  The boolean parameter `debug` activates additional messages for debugging purposes similar to the server debugging mode described in Section 3.10.2. The parameter's default value is `false`.

[12]The parameter was introduced for use on those Windows systems that do not recognize the loop-back address `127.0.0.1`. Note that Windows systems of the 2000/XP family can handle it.

7.3 Audio Client

The audio client Applet can play both the older WWR2 and the new WWR3 audio format described in Section 5.1. A WWR2 input stream is decoded with E-Chalk's ADPCM to an 8-bit, 8-kHz μ-law mono audio stream, which can directly be played by the Java audio system, irrespective of the underlying Java version. With a WWR3 stream, the data decodes to a 16-bit, 16-kHz linear mono audio stream. This can also be played directly within the browser, provided is supports Java 1.3 or later. If only Java 1.2 or earlier is supported, the client converts the data to 8-bit, 8-kHz μ-law to replay them. This means that users with a more recent Java version can listen to the audio at a higher quality than others. The recording data and the client used are the same for both types of users.

To compensate for network jitter, the audio client uses a buffer. Filling the buffer with data introduces a delay for live transmissions. For two-way transmissions, round-trip delay would be twice that. The WWR2 implementation used a delay of 12,700 ms, while in the new WWR3 version, the buffer time was reduced to 4,763 ms. The values are results of experiments to find a good trade-off between interruption-free transmission and small delay. The smaller delay in the newer version reflects improvements in the Internet's quality of service achieved in the last five years.[13]

7.3.1 VCR Operations

The audio-data stream consists of packets that contain audio data for a fixed duration (4,763 ms). The storage size of the packets varies and is stored in the packet's header. To fast-forward WWR2 recordings, the client runs through the packet's headers until it skipped enough packets. Even worse, for jumping backwards in time, the client has to read the audio data from the beginning until it reaches the desired packet.

For WWR3 data, this method is used only as fallback. Nearly all Web servers today support HTTP requests for random-access reads [FGM+97, FGM+99]. The WWR3 server stores both the encoded audio data and an index file, which contains the sizes of the audio packets. With the index information, the audio client can jump directly to the right packet, dramatically speeding up the operation. The old method is used only if the index file cannot be accessed or if the Web server does not support byte-range requests.

7.3.2 Parameters

Parameters recognized by the audio client Applet include[14]:

- `archivemode`

  This parameter gives the location of the recorded audio data as a relative URL. If the value ends with a slash, it is assumed to be a directory that contains the encoded WWR3 audio file `content.wwr` and the packet index

[13] The delay compares well to other streaming systems. For example, streaming with the Real Presenter has a one-way delay of 30 s. Even satellite phones have a delay of about one to two seconds [ZS02].

[14] Parameters that are supported only for backwards compatibility are omitted.

file `index.wwr`, see Section 7.3.1. Otherwise, the parameter is assumed to be a WWR2 audio file (or a WWR3 audio file if `wwr3mode` is set to `on`, see below). When this parameter is undefined, the client assumes a live-stream connection. For historical reasons, the parameter `ondemandmode` can be used instead of `archivemode`.

The default setup of the E-Chalk system sets the `archivemode` parameter to `audio/` for archived lectures. When the WWR2 format was still in use, the parameter was set to `lecture.wwr`.

- `port`

  The parameter is the decimal port number to connect to on the live server. When not given, it defaults to port 9998. The parameter is ignored when replaying from archive.

- `loopmode`

  With the audio system originating from an Internet radio streaming system, the audio Applet terminates by default when reaching the end of the audio stream, releasing all system resources. The audio client terminates at the end of replay and cannot be played back. To avoid this, the parameter `loopmode` has to be set to `on`, being basically inverse in effect to the board's `autoclose` parameter.

- `syncalpha`

  A problem when recording audio is that sound cards do not return the audio samples at a precise enough rate for our synchronizing purposes. Even for high-quality sound cards the actual duration of the audio sample usually differs from the requested duration by a few seconds per hour of recording. The audio client determines its time offset by counting the audio samples, while the other streams use the system clock. With an average sound card used for recording, the differences may add up to about 30 seconds for a lecture of 90 minutes, and the offset may become quite irritating at the end of a longer recording.[15]

  To get decent synchronization between the board and audio streams, the offset time computed by the audio stream is multiplied by a correction factor α. This value is determined by the E-Chalk application at the end of the recording as T_{board}/T_{audio} where T_{board} and T_{audio} are the total times of the board and audio recordings. (This is done similarly for recording audio and video without the board stream.) The `syncalpha` parameter is set accordingly in the default templates, stored as a string representation of a positive double value.

- `wwr3mode`

  When set to `on`, the audio client assumes the input to be in WWR3 format, even if the `archivemode` parameter is a non-directory file (not ending with a slash), see above. The parameter must also be set in live streaming mode (i. e. when the parameter `archivemode` is undefined) if WWR3 data are streamed.

[15] For live transmissions, this problem does not occur since the replay time is determined by the receiving time.

Figure 7.7: A video of the instructor replayed in the video client Applet.

- `server`

  This parameter is identical in function to the board Applet parameter of the same name. It is used for debugging purposes only.

- `errorurl` and `endurl`

  The parameter `errorurl` defines a URL to redirect the browser to if the connection is down or too slow. In Internet radio, this was used to either redirect to an alternative broadcaster or to change to an HTML page with the audio client for a connection serving at a lower bandwidth. The latter approach allows to have a set of streams with different bandwidth where the client automatically selects the highest bandwidth the listener's connection can handle.

  The parameter `endurl` was also only used for Internet radio. If set to a URL, the browser changes to the given location when the client terminates on reaching the end of the stream.

7.4 Video Client

As stated in Section 5.3 in the description of the video format, the E-Chalk video is intended more to add a personal touch to recorded lectures than to be a major source of information. As high-resolution video consumes lots of bandwidth, the videos recorded with E-Chalk are normally of low resolution. See Figure 7.7 for an example.

The buffering strategy of the video client the differs from the one in the audio client, as many client systems may not have enough memory available for buffering video for several seconds. For video, a dynamic cache is implemented, that increases and decreases with the amount of memory available. Fortunately, discontinuous image streams are not as disturbing as discontinuous audio streams.

Jumping forward and backwards in recorded video is by now handled as in the WWR2 audio. The faster approach of VCR operations in WWR3 audio, using the index file to jump to the proper offset in the video data file, is planned as a future enhancement. Since video encoding uses only difference frames (except for the very first frame), jumping in time generates artifacts in areas which changed only in the skipped time. This can be avoided by storing full frames (I-Frames) every few seconds in separate files, another feature planned for the recent future. In combination with the index file, this allows the client

fast random access and to fully construct the video by loading the most recent I-Frame. While the storage of the I-frames would increase the size of the stored video notably, only a single I-frame per jump has to be loaded, keeping the low bandwidth requirements for the remote user.

Parameters recognized by the video client Applet are:

- `archivemode` (or `ondemandmode`)

  Similarly to the audio client parameter of the same name, this gives the URL (relative to the code base) of the directory containing the encoded video `content.wwv`. If the parameter is not set, the client works in live-streaming mode.

- `delayoffset` and `masiclient`

  These parameters are identical to the board client parameters of the same name, used for synchronization with the audio client, see Section 7.2.5.

- `server`, `loopmode`, and `port`

  These parameters are identical in function to the parameters of the same name for the audio client, see Section 7.3.2. The only difference is that the default value for the live port is `9997`.

7.5 Slide Show

The slideshower Applet allows timed display of Web pages and to combine it with the other client Applets shown before. The slide show in Figure 7.8 combines a sequence of pages with the replay of a talk.

The implementation of the slide-show Applet does not support streaming of slide display command. The slide events are loaded on startup before the replay starts and thus live transmission is not supported. While the other client Applets can be run with the Java `appletviewer`, the slide-show Applet needs a browser environment, as the `appletviewer` cannot display Web pages.[16]

With no live server currently available for the slide-show Applet, this client currently supports only archived replay. It causes the browser to load URLs at given timestamps. The data for the slide show are a text file containing one entry per line (irrespective of the platform-specific encoding standard different line separators may adhere to). A command to display a document is

`showurl`␣*url*␣*hh*␣*mm*␣*ss*[␣*target*]

where *url* is the URL of the document to be displayed, and *hh*, *mm*, *ss* are the display time as hour, minute and second offset to replay start time, given as two-digit decimals. The optional field *target* specifies the frame to open the document in either the name of the frame in the HTML or one of the symbolic targets

[16] The `java.applet.AppletContext` method `showDocument(URL, String)` is used to show the pages. According to the Java API documentation, browsers are free to ignore the calls, see Java API [42]. In practice, however, all Java-capable browsers can be assumed to support this method.

As a side note, these document to be shown need not be stored locally in contrast to the images and Applets loaded by the board client. As the slide-show documents are loaded into the browser and not into the Applet, the usual Applet security restrictions do not apply to these resources.

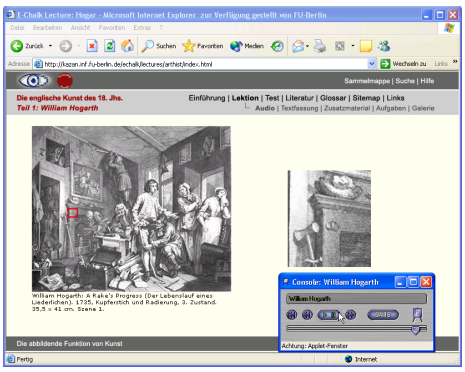

Figure 7.8: A snapshot from [23], a slide show with audio and control panel created by *Schule des Sehens* [86], an e-learning project on art. The different lesson slides illustrate different details of the art work explained.

`_self` the frame, that contains the slide-show Applet, `_parent`, the Applet's parent frame, `_top`, the top-level frame of the Applet's window, and `_blank`, a new unnamed top-level window.[17] If a named frame is used and a frame of that name does not already exist, a new top-level window with the specified name is created and the document is shown there. If the parameter is omitted, the default target is used, initially set to the target name `slideframe`. The `showurl` commands are executed in the order of their timestamp values. When two commands use the same timestamp, they may be executed in any order.

The other type of entry is used to set the default target for following `showurl` lines. Its syntax is

`defaulttarget`[␣*target*]

where *target* is the new target. If it is omitted, the default target is set to the frame with the name `slideframe`, creating it in a new top level window, if it does not yet exist.

The data file for the slide show can be created either directly with a text editor or, more comfortably, with the Exymen editor, see Section 6.13.

Jumping to a timestamp t_0 is realized as displaying the last document to be shown in the interval $[0, t_0]$.

The parameter recognized by the slide-show Applet are:

- `slidefile`

  This mandatory parameter gives the (relative) URL of the slide-show command file to be used.

- `delayoffset` and `masiclient`

  These parameters are identical to the board-client parameters of the same name, used for synchronizing with the audio client, see Section 7.2.5.

[17] See [RLJ98] for target names.