Chapter 9

Outlook

Much good work is lost for the lack of a little more.

Edward H. Harriman

The system has already been extensively evaluated in university settings and, to a lesser extent, in K-12 schools. For other application scenarios, however, an evaluation study has not been conducted yet. Also, the impact of the new audio system on the perceived quality still has to be systematically validated.

As mentioned before, a number of possible future improvements have already been identified, like a more intelligent page-breaking algorithm in PDF production and many other useful improvements, for example providing board backgrounds with lines, grids, and logarithmic chart-paper-style patterns. Beyond minor features to be added, research is desirable on the following points.

Methods for Keyboard Input on the Board

The need for using keyboard in the board environment should be completely eliminated. An obvious solution would be to integrate a general handwriting recognition feature. However, reaching a satisfying level of recognition reliability is difficult unless the user is required to learn a special alphabet, like *Unistroke*.

Integrating a software keyboard is not an adequate option since operation on an E-Chalk board is still too awkward. Instead, using hierarchical pie menus[1] may be a more viable option. See Section 1.4 for a description of employing pie menus for key input.

Transmitting the Board Pointer

A feature sometimes requested by instructors and learners alike was a pointer or semi-transparent marker tool for the board. Some instructors use the system's mouse pointer for referencing. While visible in the classroom lecture, the action is not stored for remote access. However, having to operate with a mode for pointing would put an extra burden on the instructor. Very likely the

[1] Pie menu variants implementing hierarchical organization, called marking menus, are protected by patent [KF98]. They are often used in gesture-based interfaces. The menu is only displayed for beginning users and kept invisible for expert users, who have already learned the stroke movements for different options.

teacher would often forget about the tool. The developers of *Classroom Presenter* reported, for example, that in practice none of their instructors found a semi-transparent highlighter useful [AAS+04].

A better approach is to record the mouse pointer all the time. Displaying it on the client side would fit into the philosophy of delivering the same information to the learner in the classroom and to the remote user. It does not require the lecturer to adapt his or her teaching to the technical requirements of the remote viewer. Constantly capturing the mouse pointer position results in extra data to be transmitted, but the data volume in question does not exceed the levels reached for drawing events. As shown in the bandwidth analysis in Section 4.10.1, the actual data volume is negligible given the event rates delivered by standard pointing devices.

Chalklet Support

The chalklet concept is a quite recent addition to the E-Chalk system and therefore the API library for chalklet developers and the underlying chalklet management are at an early stage of development.

In the future, chalklet code should be restricted using Java security mechanisms like the `java.lang.SecurityManager`. The execution of chalklets should be secured by concepts similar to Java Applet execution is in browsers, running them in a secure "sandbox". This would protect the main E-Chalk application against malignant chalklets, for example preventing chalklets from terminating the whole application by calling `System.exit(int)`. However, some restrictions posed on Applets like prohibiting file access and very restricted network access might turn out to be too restrictive for productive chalklet programming. This will have to be examined in detail.

Changing the uncompressed ASCII board event format to a binary format with differential encoding would allow to relax the limits of maximum event rates on chalklets (and macros) without running the risk of exceeding bandwidth limitations, see Section 4.10.1.

"Functional chalklets" may even be created so as to encompass existing drawings on the board and then perform particular actions on these drawings or even on the output of previously executed chalklets.[2]

To assist in chalklets development, the supporting API should be extended. Methods are to be provided to conveniently create `Stroke` objects for basic geometric shapes (boxes, circles, etc.) and for printing texts, perhaps even in a handwritten style.[3]

Ideally, a recognition engine for strokes should become part of the API to support complex interpretations of stroke input to chalklets. Geometric interpretation of freehand sketching is already an active research area, often in the context of pen-active whiteboards with office-type applications. For example, the *SATIN* [HL00] Java toolkit is a framework for pen-based applications which processes both stroke objects and stroke gestures. Example applications built on top of *SATIN* are *DENIM* [LNHL00] for building Web pages by sketching, *SketchySPICE* [HL00] as a simple-circuit CAD tool and *SILK* [HLLM02] for prototyping user interfaces by sketching.

[2] This feature was suggested by [Wat04].

[3] See handwriting synthesis description in Section 6.11.

At MIT, another framework for sketch recognition was developed [Sez01, HD04], including a description language for drawings called *LADDER* [HD03]. Applications realized with the framework include *Tahuti* [HD02] for creating UML diagrams by sketches and *ASSIST* [AD01], a sketch-based CAD system.

Another example of a stroke-based interface is *Flatland* [MIEL99, MIEL00], described in Section 1.5.4.

Improvements to Replay

In addition to the standard VCR operations provided for replay, it would be useful to provide a kind of spatial control for the board stream. A scaled-down version of the final board content could well be provided. The learner should then be able to jump to the time offset at which selected board content was created. This would enable users to navigate in the lecture to a subject without having to search along the time line.

Small-screen rendering techniques should be applied to enable replay on platforms that offer only resolutions lower than the one used for the recorded board, see Section 8.1.3. Possible approaches include automatic scrolling to currently changed content and fish-eye-view techniques. This would enable replay on hand-held devices as well as recording with high-resolution hardware.