

Chalklets: Developing Applications for a Board Environment

Lars Knipping
Berlin University of Technology
knipping@math.tu-berlin.de

Marcus Liwicki
University of Bern
liwicki@iam.unibe.ch

Abstract

This paper presents a novel software framework and methodology to run applications on an interactive whiteboard. A new type of application is created that is called “Chalklets”. Chalklets use the chalkboard as interface metaphor instead of the desktop. This allows the seamless integration of educational mini applications into electronic whiteboard-supported lectures. This article describes the framework, presents a number of Chalklets, and discusses the idea using the example of a Chalklet for simulating sketched logic circuits.

1 Modern use of the Chalkboard

A number of lecture recording and streaming systems has been developed in the past years to allow easy creation of e-learning materials from everyday teaching. The most direct approach is to use mere video capturing of the lecture. However, this approach needs technical personnel present during the recording to handle the camera and the audio hardware. Furthermore, encoding with off-the-shelf Internet video tools is often inadequate for lecturing content because writing and drawings, from slides or from a blackboard, are not encoded appropriately, which leads to artifacts and loss of semantics.

One alternative is the use of lecturing recording systems using desktop capturing, for example commercial products like *Camtasia Studio* or *Tegrity*. Typically, these are employed to capture talks held with slideware tools like PowerPoint, as the personal desktop itself is rarely suited for teaching purposes.

Other tools add whiteboard functionalities to allow annotations and excursus with freehand drawings and writing. An early example is *AOF* (authoring on the fly), a system for both synchronous and asynchronous delivery of lectures. Transmission includes audio and whiteboard activities, usually annotated slides. Video and animation applications can also be added [4].

Classroom2000, later renamed to *eClass*, records lecture slides and handwritten annotations as static Web pages together with audio and (optionally) video recordings of the instructor as well as any Web page he or she visited during the lecture [1].

The platform independent *Transparent TeleteachingTool* (TTT) combines desktop recording including whiteboard functionality with audio and video from the lecturer [17].

The *ConferenceXP Presenter* or *Classroom Presenter* [3] is used for both classroom teaching and synchronous distance teaching. The presenter organizes content in slides to be annotated. To provide a kind of whiteboard functionality, the instructor may insert empty slides to be filled with freehand writing and drawings.

We decided to use a more radical approach towards freehand input. Even though slideware tools can be used to give a well-structured and easy-to-follow lecture when correctly employed, they foster a tendency to overwhelm learners with an overly rapid presentation of information [6]. The lecturers, of course, possess a deeper understanding of the subject and often tend to progress through the lecture at a pace too fast for their students to follow.

Investigating established teaching techniques, one finds that the old-fashioned chalkboard is still unsurpassed as a teaching tool due to its intuitive use and unmatched flexibility. The learners can see how ideas are developed rather than being overwhelmed with final results: they are supported in following the conceptual process. The process of handwriting imposes a natural limitation on the pace of the lecture, giving the students time to follow the lecturer’s train of thought. Compared to the use of prepared slides, the “chalk and talk” approach results in a much more flexible teaching style. Working on a chalkboard supports creative thinking, illustration, and sharing. Board drawings can be used to draw attention to details using circles, arrows, underlines, checks, groupings, etc. Thus it comes as no surprise that the chalkboard is still so popular

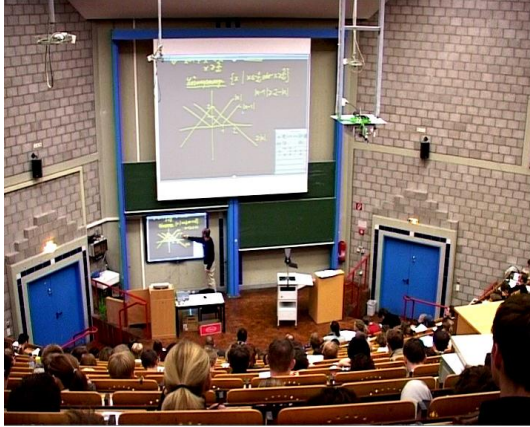


Figure 1. Mathematics lecture at Berlin University of Technology using a digital whiteboard model and an additional projection for the audience

for teaching in many disciplines, especially for subjects where complex reasoning has to be taught, such as mathematics, engineering, and the natural sciences.

These considerations inspired the development of a system called eChalk [6]. During classroom teaching, the lecturer works directly on a pen-active wall display, see Figure 1. Ideally, lecturers are enabled to teach with the system just as with a regular blackboard.

The system relies on the chalkboard as user interface metaphor which is more appropriate here than the usual desktop metaphor. The desktop is designed as a personal workspace while a board is a workspace to be shared with an audience and serves as a common focus point for observation and discussion.

Keeping the ease of handling of a traditional board is quite important. In the given setting the lecturer has to concentrate on giving the lecture and must not be distracted by technical requirements. Also, the need of keyboard input is very disruptive when working with a pen at the board and should be avoided as far as possible.

All actions on the board are tracked. The development of the board content can be viewed by a remote learner, both as a live transmission or as an asynchronous replay. The voice of the lecturer can also be recorded. The distance learner is provided with a live script where teacher's side notes are not lost. These two data streams already capture most of the substance of the lecture. Optionally, a video stream of the instructor can be added to provide a more personal touch to the remote lesson.

Due to the board stream using a vector representa-

tion, the bandwidth requirements are very low. The bandwidth of the board stream peaks in the range of 3 to 5 kbps when using standard pen or mouse devices, i. e. with sampling rates of between 50 and 125 Hz. In fact, average bandwidth needed in real lectures is even less than 1 kbps [11].

Thus in practice the board's bandwidth requirement is negligible compared to the bandwidth used by audio and optional video. Let alone, for the audio stream codecs between 24 and 256 kbps can be chosen.

1.1 Teaching Elements on a Chalkboard

While eChalk simulates a classic chalkboard, an obvious idea is to enhance lecturing with it by a range of multimedia applications. These offer the opportunity to enrich the lessons, allowing the system to surpass the didactic potentials of the traditional chalkboard.

For this reason, eChalk offers several ways to incorporate images from a local disk, from known web URLs or via a search facility like Google image search. CGI-based web services delivering text or images can be called and computer algebra systems running in the background can be queried for numerical results or function plots. Java Applets can be incorporated on the board.

However, the more sophisticated these elements are, the less they fit into the board metaphor. The requirement of the keyboard can be eliminated to some extent using handwriting recognition. But the rich interactivity of an Applet cannot be transmitted anymore using only a low-bandwidth vector representation.

For this reason, a new kind of application, based exclusively on board strokes was introduced to the system. The user can start a so-called Chalklet, that can draw strokes on the board and receive user drawn strokes as input. A very basic example of a Chalklet is one that allows to play a game of Tic-Tac-Toe against a computer opponent. After a small play field is drawn, the user can draw a cross. The computer then answers with a circle. The turns are repeated until either all the fields have been filled (no winner) or one of the opponents achieves to set three crosses (or circles) in a row. This example shows the intuitiveness and simplicity of the Chalklet approach.

A number of systems using sketched input for other types of applications have already been developed. For example DENIM [14] allows to build web pages by drawing, SketchySPICE [10] is a simple-circuit CAD-tool, Tahuti [8] is used for creating UML diagrams by sketches, and ASSIST [2] is a sketched-based CAD tool.

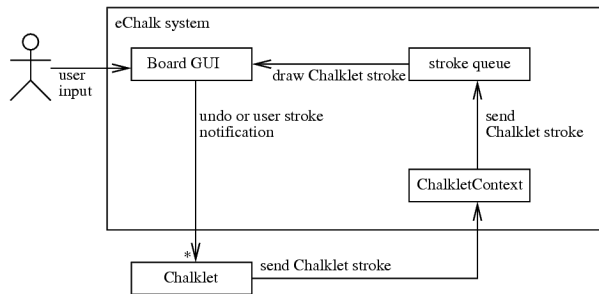


Figure 2. Communication between the eChalk board and active Chalklets

2 The Chalklet Framework

2.1 Embedding Environment

Chalklets, as the name implies, are very similar to Applets, in the sense that they are small Java applications dynamically embedded in a host application. Like Applets, a Chalklet operates in a rectangular region dedicated to it. A Chalklet is notified of any user drawings within that region and can itself send strokes to be drawn in the region. Applets can communicate to its hosting application through an instance of `AppletContext`. Similarly, a Chalklet has access to a `ChalkletContext`. This context provides run-time information such as the position of the Chalklet, the background color of the board and provides methods for sending and receiving strokes.

A stroke encapsulates the data of the polygonal chain of its sampling points, the sampling timestamps, the thickness and the color of the stroke. As the board interface also allows undo actions, Chalklets are also notified of any undo of a received strokes. A redo of a strokes is simply handled by resending the stroke. Strokes are put into a queue of strokes to be drawn, see Figure 2. Undo requests are not available for Chalklets. Deletions of drawn features are realized by simply overdrawing them in the board background color.

To avoid conflicts between strokes to be drawn in the same time interval, the queue uses an ordering according to their final timestamp. Interlacing strokes would complicate the order to be used for undoing the strokes.¹ Using the finishing timestamp instead of the starting one prefers fast strokes over slow ones, lowering the risk of the queue being blocked by slow strokes.

¹Note that the assumption of a single operating user underlies the described eChalk system. In CSCW tools like shared whiteboards there would be other mechanisms handling concurrent drawings.

A user drawn stroke, however, always takes precedence over Chalklet strokes, even if it splits a stroke into two parts. As a result, Chalklets can never block a user from painting and leaves him or her into complete control of the system.

2.2 Life-cycle

The classes belonging to a Chalklet are embedded in a jar-file for deployment and installation in the eChalk System. A programmer has to inherit from a certain class and implement a Chalklet factory that instantiates a new Chalklet.

When a user calls the creation of the Chalklet, he or she is asked to select a region on the board for it. The Chalklet classes are then loaded dynamically and are constructed immediately. The Chalklet is then able to send strokes or wait for user entered strokes to react to.

When the board area is scrolled and an active Chalklet leaves the visible viewport, the Chalklet is deactivated. It does no longer receive any input, its pending strokes in the stroke queue are removed, and the Chalklet instance itself is notified of its deactivation, so it can release any resources it still holds. This forced deactivation prevents a Chalklet from cluttering the board with its output while being unobserved.

In summary, creating and operating a Chalklet involves the following steps:

1. The user selects a Chalklet factory.
2. The factory returns the minimum area size needed.
3. The board lets the user select a Chalklet area.
4. The board creates a `ChalkletContext`.
5. The factory creates a Chalklet for the given context.
6. The Chalklet is active and able to send and receive strokes.
7. The Chalklet terminates or is stopped by the board.

2.3 Guaranteeing Bandwidth

As another problem to address, a Chalklet may easily create strokes several magnitudes faster than user drawings can and vastly exceed the low bandwidth requirements the board stream otherwise has. In fact, by painting each pixel of a video frame with a small single-pixel stroke, it would even possible to show an arbitrary video animation.

To enforce the low bandwidth usage and a board-like look and feel, an additional restriction is introduced into the stroke processing mechanism: The strokes are not processed faster as a typical pen or mouse input device would produce them. The stroke drawing mechanism operates at a rate of 100 Hz, i. e. up to one sam-

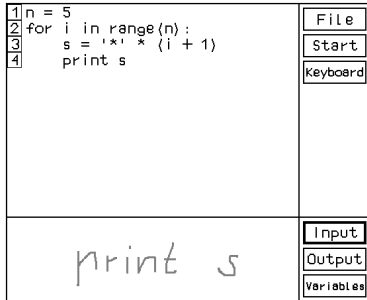


Figure 3. A Python interpreter Chalklet

pling point is processed per ten milliseconds. Applying this constraint, the bandwidth bound mentioned in Section 1 still holds.

3 Sample Chalklets

A number Chalklets has been developed within the presented framework. Some of them are briefly presented in this section in order to provide evidence for the potential that lies this simple approach.

Animated sorting and graph algorithms have been realized as Chalklets [5]. These are aimed at visualizing algorithms in computer-science teaching. The Chalklets take user strokes as hand-drawn input to the chosen algorithm, e.g. a collection of strokes to sort according to their height, or hand-drawn graphs, using the lengths of the connecting edges as weights.

A more sophisticated application, a solution for simulating biological and pulse-coded Neural Networks is presented in [12].

A Python interpreter Chalklet was developed [16]. The technical overhead from the Python language syntax is very small compared to most other languages and programs look very similar to algorithms in pseudo-code notation. The idea of building a Python-interpreter Chalklet came in mind when computer-science lecturers were looking for a tool to interactively teach programming and algorithms with eChalk. The Python Chalklet supports interaction types of a simple Python programming environment interface to be used during a lecture, see Figure 3. It accepts handwritten Python commands that are then recognized by the Chalklet. Recognized writing can also be inserted at arbitrary program positions, again by a drag-like gesture, and recognized letters can be deleted by a strike through. As a fall-back input possibility, the Chalklet provides a keyboard-like input. It shows all letters and takes any letters stroked out as input.

3.1 A Logic Circuit Simulation Chalklet

In the following, the realization of a larger application is presented as to show the feasibility of the Chalklet approach also for more complex solutions. The example also demonstrates many of the methods typically used for implementing Chalklets.

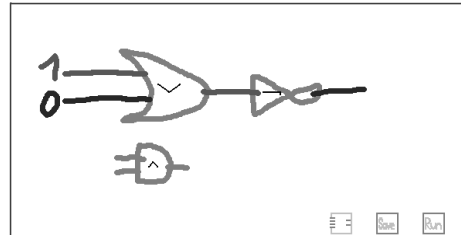


Figure 4. A circuit simulator Chalklet

The Chalklet presented here is a graphical simulator for logic circuits [15]. The circuits are developed as freehand drawings, see Figure 4. The application was developed for use in computer-science teaching. Note that earlier simulation engines rely on drag-and-drop interfaces or even textual circuit definitions [9, 13].

3.1.1 Using the Chalklet

The application recognizes the symbols for the logic gate types *and*, *or*, *multiplexer* and *demultiplexer* (shown in Figure 5) as well as wire connections between them. The user is expected to start drawing symbols from one of the ending points at the input side of the gate², and the symbol can be drawn into any of the four axial directions, i.e. the input side can be on the left, at the top, at the bottom or on the right.

A predefined color is reserved for drawing the wires and gates. The system detects the type of the elements and gives a visual feedback at the center of its bounding box, see for example Figure 4. Each gate can be used with an arbitrary number of input and output lines. The leftmost of the three buttons at the bottom of the panel shows the number of input and output connections of the entire circuit, see Figure 4. For a description of the recognition process see Section 3.1.2.

For simulating the sketched logic circuit there are two possibilities. First, a simulation can be started by drawing a one (high) or a zero (low) in a color reserved for logic level input. The system then sets the nearest input of the circuit to the corresponding state. The

²This restriction is used to distinguish between a multiplexer and a demultiplexer oriented in the opposite direction, which look identical.



Figure 5. Gates recognized by the application: and, or, not, multiplexer, demultiplexer

result is visualized by repainting the strokes of the input level and the wires in a color for the corresponding logic level. As described in Section 3.1.3 the states of all connected wires are updated immediately.

The state of an input can be changed anytime by drawing a new input value. Alternatively the button with the caption “Run” shown in Figure 6 may be activated by drawing a stroke inside the button area. All possible combinations of the uninitialized input lines of the circuit are then simulated sequentially.

The simulator also allows the user to synchronize the gates with a timer. A clock symbol, drawn in a third reserved color, can be connected to the desired gates of the circuit, see Figure 6. The connected gate retains its output until one clock cycle elapsed. A clock can be started and stopped by stroking on the clock symbol.

If the user touches a previously sketched element with a background colored stroke, the element is completely removed by the system. This serves as a convenient delete mechanism for the user.

Figure 6 also illustrates another feature, namely displaying a state-timing diagram. For every clock cycle the states of the input and output lines of the circuit are illustrated in the diagram. If no timer is present all circuit states are added to the diagram for each change of the circuit input. For the sake of clarity, the input and output lines are numbered from top to bottom in the state-timing diagram. See for example the RS-Flip-Flop shown in Figure 6.

It is also possible to save a circuit with a self-defined symbol and to reload it in future sessions. A symbol of the circuit can be drawn in the input and output displaying box mentioned above, see Figure 6. By activating the button labeled “Save”, the graph and the symbol of the circuit are stored into a repository of circuits. In later sessions it can be loaded by drawing a rectangle in place of the stored sub-circuit with a fourth reserved color. All symbols of the repository are then drawn in the right hand side of the Chalklet area. The user selects the circuit by stroking in the corresponding symbol. The system then marks the drawn rectangle with the symbol and the input and output connections. Figure 7 illustrates the integration a previously defined RS-Flip-Flop into a circuit.

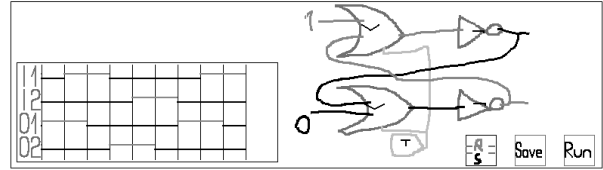


Figure 6. RS-Flip-Flop with a clock symbol and with a state-timing diagram

3.1.2 Recognition Engine

In a first step, the stroke recognition decides whether an input stroke represents a logic gate or a wire. Gates are assumed to be closed, i.e. the distance between start and end of the stroke is small.

For wires, three different kinds have to be distinguished: input, output, and connector. This classification depends on whether it touches the input or output side of a gate. Any connection via a touching wire is recursively considered.

To classify a gate-type input, a multilayer perceptron network is applied after some preprocessing to normalize the input data. Most preprocessing steps are adopted from [7]. Please refer to [15] for details.

3.1.3 Simulation

An internal graph representation of the circuit is created for the simulation, similar to the one used in [9]. The edges of the graph correspond to connection points of wires and gates. For a connection of gates and wires the direction of the signal flow is defined by the direction of the gate symbol. For all other connections the direction is calculated recursively by using the information of the neighboring wires. This direction determines if a wire with a unconnected end serves as input or output to the circuit.

Any new circuit input is transmitted through to any neighbor until no element changes anymore and a stable state is reached. The state of a wire is set to undefined if no stable state exists.

4 Limits of the Approach

The aim of having an interactive application fitting into a board style environment does not well match with fancy high-resolution animations or video shows. However, it is quite feasible to realize useful teaching tools even with the given constraints. Interactive versions of demonstrations shown with a chalk-and-wipe approach on a traditional board are easily trans-

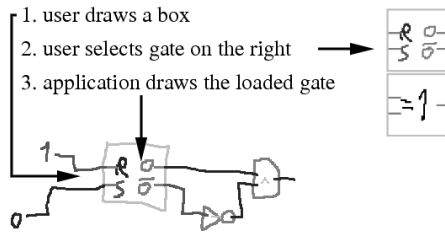


Figure 7. Loading an RS-Flip-Flop saved before into the application

lated into Chalklet applications. Areas where highly symbolic handwritten notations are used are especially well suited, as such symbols can be easily produced and recognized by both human users and the Chalklet.

Nontrivial applications for Chalklets must usually feature advanced recognition abilities to be able to handle a wide variety of user input. A generic recognition engine for freehand input as part of the Chalklet API would be a great help in developing further applications.

5 Summary

In this paper we presented a software framework and methodology for developing applications to be run in an interactive whiteboard environment. The input and output methods of the new application type called Chalklet are based on board strokes exclusively. This allows the seamless integration of educational applications into whiteboard supported lectures.

The paper showed the use of pen strokes as sole communication method being sufficient to create complex Chalklets with a large functionality. As a proof of concept a logic circuit simulation Chalklet is presented, which has been developed for use in university and school education. In this, circuit gate symbols and connecting wires are recognized. The simulation is fully controlled by hand-drawings.

Due to the ability of allowing the integration of applications with a large functionality, this framework can be useful for whiteboard-supported lectures, even though the control is restricted to board strokes. However, this simplicity allows an easy adaption of the framework into other stroke-based systems.

References

[1] G. D. Abowd. Classroom 2000: An experiment with the instrumentation of a living educational environ-

ment. *IBM Systems Journal, Special Issue on Pervasive Computing*, 38(4):508–530, October 1999.

[2] C. Alvarado and R. Davis. Resolving ambiguities to create a natural computer-based sketching environment. In *Proceedings of IJCAI*, August 2001.

[3] R. Anderson, R. Anderson, B. Simon, S. A. Wolfman, T. VanDeGrift, and K. Yasuhara. Experiences with a tablet PC based lecture presentation system in computer science courses. In *Proceedings of the 35th SIGCSE technical symposium on Computer science education*, pages 56–60, March 2004.

[4] C. Bacher and T. Ottmann. Authoring on the fly. *Journal of Universal Computer Science*, 1(10), October 1995.

[5] M. Esponda Argüero. *A New Algorithmic Framework for the Classroom and for the Internet*. PhD thesis, Freie Universität Berlin, August 2004.

[6] G. Friedland, L. Knipping, J. Schulte, and E. Tapia. E-Chalk: A lecture recording system using the chalkboard metaphor. *Interactive Technology and Smart Education*, 1(1):9–20, February 2004.

[7] W. Guerfali and R. Plamondon. Normalizing and restoring on-line handwriting. *Pattern Recognition*, 26(3):419–431, 1993.

[8] T. Hammond and R. Davis. Tahuti: A geometrical sketch recognition system for uml class diagrams. In T. Stahovich, J. Landay, and R. Davis, editors, *Papers from 2002 AAAI Spring Symposium on Sketch Understanding*, pages 59–66, March 2002.

[9] N. Hendrich. HADES: The Hamburg design system. In *EASA '98/Alt-C Conference: Lifelong Learning on a Connected Plane*, September 1998.

[10] J. I. Hong and J. A. Landay. SATIN: a toolkit for informal ink-based applications. In *Proceedings of the 13th UIST*, pages 63–72, November 2000.

[11] L. Knipping. *An Electronic Chalkboard for Classroom and Distance Teaching*. PhD thesis, Freie Universität Berlin, February 2005.

[12] O. Krupina. *Client-Server Architecture for a Neural Simulation Tool*. PhD thesis, Freie Universität Berlin, July 2005.

[13] L. Li, H. Huang, and C. Tropper. DVS: An object-oriented framework for distributed verilog simulation. In *Proceedings of the 17th PADS*, June 2003.

[14] J. Lin, M. W. Newman, J. I. Hong, and J. A. Landay. DENIM: finding a tighter fit between tools and practice for web site design. In *Proceedings of CHI*, pages 510–517, April 2000.

[15] M. Liwicki and L. Knipping. Recognizing and simulating sketched logical circuits. In *Proceedings of the 9th KES Conference, Part III*, September 2005.

[16] H. Steffien. Handschriftliche Erstellung und Ausführung von Python-Skripten auf der E-Kreide Tafel. Bachelor's thesis, Freie Universität Berlin, September 2004.

[17] P. Ziewer and H. Seidl. Transparent teleteaching. In A. Williamson, C. Gunn, A. Young, and T. Clear, editors, *Proceedings of the 19th ASCILITE*, pages 749–758. UNITEC Institute of Technology, Auckland, New Zealand, December 2002.