Chapter 9

NeuroSim as an intelligent E-Chalk assistant

As a further step of development, NeuroSim can be used in conjunction with other programs as an effective tool for educational purposes. In principle, NeuroSim can be started during a lecture as an application or as an applet directly from the web. However, a more effective solution would be a direct integration of NeuroSim into modern teaching systems, such as the electronic chalkboard E-Chalk.

The E-Chalk system preserves all the functionalities of a traditional blackboard as an effective teaching tool (see Fig. 9.1). At the same time, E-Chalk as the digital system makes it possible to extend a traditional lecture with the capability of handling multimedia elements. It can for example integrate pictures directly into the board environment or start certain programs or even applets on demand. Last but not the least, distance lectures can be produced from ordinary classroom teaching[57].

The E-Chalk system can be extended with special programs known as "intelligent assistants", which react to board drawing elements, simulating "intelligent" responses on the actions of the lecturer. NeuroSim has been integrated into E-Chalk as such an application, allowing the teacher to draw neural elements directly on the board and simulate their functioning right on the board.

I rejected the idea of loading NeuroSim as an applet in E-Chalk for the following reasons. First, using NeuroSim as an applet within the E-Chalk environment would not preserve the main feature of E-Chalk. The interface of an applet contains such elements as windows, pop-ups or menus, which do not correspond to the natural environment of a board. Second, the safe functioning of applet applications is still under development in E-Chalk, making it difficult to ensure stable functioning of the applet version of NeuroSim.

In the first part of this chapter, I describe the main features of E-Chalk. The most important functions of the E-Chalk API concerning the development of "intelligent assistants" are outlined. Then, the handwriting recognition sys-

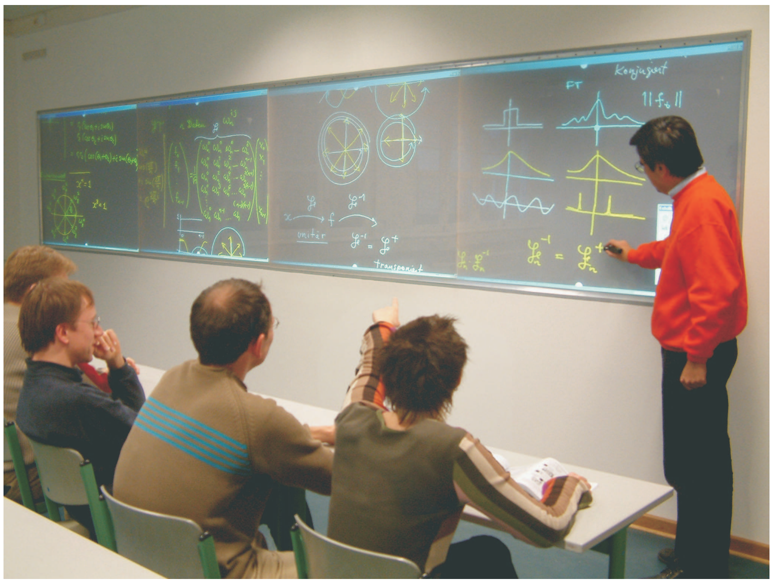


Figure 9.1: A classroom equipped with E-Chalk.

tem used in the “intelligent E-Chalk assistant” NeuroSim is reviewed. Finally, NeuroSim, as an integrated part of E-Chalk, is presented and the GUI adapted for E-Chalk is shown in use with two example models.

9.1 The E-Chalk System

9.1.1 Features of E-Chalk

The most common way of extending the presentation of material to an audience during a lecture is by using a slide show, e.g. a Power Point presentation. The material for such presentations is prepared in advance. The advantage of using slides is that a lot of information can be prepared to be shown in a short period of time, including all details and with a well thought-out design. The disadvantages of slides are that the sometimes big amount of details often dispels the attention of the auditorium from the main aspects and that the course of the lectures is fixed, not allowing any deviation from the predefined outline. It may be difficult for the lecturer to keep a balance between the amount of information given during a lecture and the tempo of its presentation, to give time for the auditorium to follow it and take notes.

In contrast, in the case of the traditional blackboard, the material is presented in parallel with the explanations from the lecturer in direct contact with the auditorium. Since the lecturer has to present the material on the blackboard ”live”, he naturally starts with the most important information and later proceeds to details. He can thereby account for the reaction of the auditorium by changing the course of the lecture, giving or excluding some aspects of the considered material. The tempo of the lectures is also naturally synchronized with the auditorium.

The E-Chalk environment was developed with all functions of the traditional blackboard providing the "look and feel" feature. It also provides features to be used for distance learning. All of the material presented at the E-chalk can be translated on-line or stored for later on-demand viewing. Anyone can follow the lectures remotely as if being present in the auditorium or reproduce the course of a given lecture later at any time.

From a technical point of view, a large contact-sensitive display, on which the lecturer can write, is used instead of the traditional chalkboard (see Fig. 9.1). Alternatively, a digitizer tablet could be used as a tool for writing, since all what is written on it can be projected with an LCD projector onto a wall. The chalk as the traditional drawing tool is replaced by a stylus [23].

It was decided to use the electronic chalkboards as the hardware component for the E-Chalk system, since it corresponds to the traditional classroom lecture, where the lecturer writes directly on a board in front of the auditorium. The large contact-sensitive display provides enough teaching place for most purposes. The electronic chalkboards also offer a wide color palette and high contrast [25].

E-Chalk lectures are considered as the future of lectures. Although the price of electronic chalkboards today is too high for most organizations to be able to buying them, one expects they will be much cheaper over the next few years. Digital systems with well thought-out interfaces and wide electronic capabilities will probably gradually substitute our conception of traditional chalkboards.

The current version of E-Chalk is implemented in Java. The lectures can therefore be viewed over the Internet with any Java-enabled browser without requirements to install extra programs. This can be done by starting an applet containing the content of the board, the voice of the lecturer and optionally a small video of the class. Fig. 9.2 shows an example of an actual lecture, given at the faculty of Computer Science at the Free University Berlin, loaded from the database of saved E-Chalk lectures.

9.1.2 Software architecture

Upon starting a lecture, the E-chalk server creates three main streams: the stream of the board context, the stream of the audio signals, and optionally a stream of the video signals. The server can be configured within the Startup Wizard menu. The architecture of the E-chalk system is shown schematically in Fig 9.3. The board content, audio and video information can be transferred into the Internet in a real time regime or be stored on the server to be provided later. The size of the transferred video is limited by the connection bandwidth. To optimize the use of the connection resources the window with video information can contain just a small piece of the lecture hall or be closed.

The E-Chalk client is responsible for the lecture transmission at the "other end of the cable". It can start three synchronized applets presenting the board

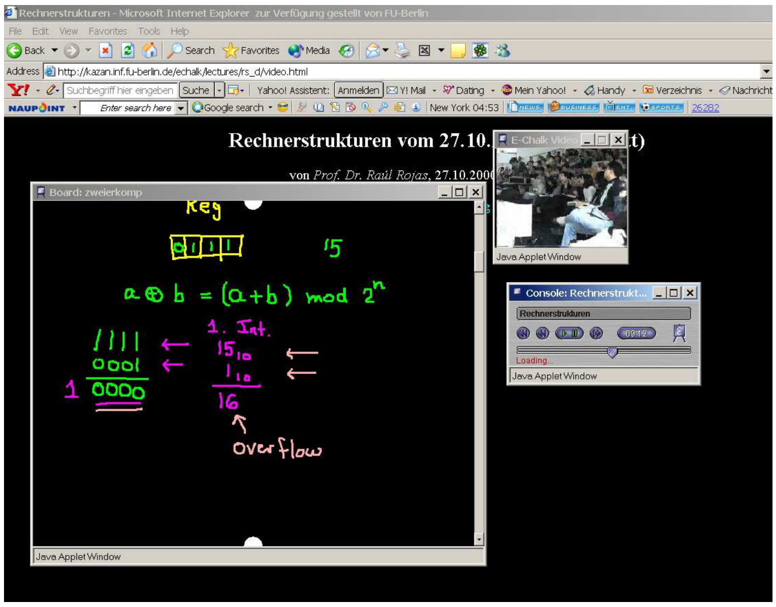

Rechnerstrukturen - Microsoft Internet Explorer zur Verfügung gestellt von FU-Berlin
http://kazan.inf.fu-berlin.de/echalk/lectures/rs_d/video.html
Rechnerstrukturen vom 27.10.
von Prof. Dr. Raúl Rojas, 27.10.200
Board: zweierkomp
E-Chalk Video
Java Applet Window
Console: Rechnerstrukt...
Rechnerstrukturen
Loading...
Reg
15
a ⊕ b = (a+b) mod 2^n
1. Int.
1111
0001
1 0000
15_10
1_10
16
overflow

Figure 9.2: Example of an E-Chalk lecture played back in a browser.

information, the audio and the video. When the user opens the appropriate URL in the browser the applets are started on the client side. The by the E-Chalk client automatically generated web pages are opened online or, in the offline case, from the server that stores the information.

9.1.3 Interface of the E-Chalk system

The interface that is provided by the E-Chalk system has been developed to keep a traditional blackboard atmosphere during the lectures.

When starting the E-Chalk system, the E-Chalk Startup Wizard that allows one to setup the properties of a new E-Chalk lecture appears (see Fig. 9.4). On the first panel the path where the lecture will be saved, overwritten or appended is determined. There are also panels for adjusting the properties of the audio, video and board streams.

After that, E-Chalk provides a one-color working area to the user (see Fig. 9.5). This working area plays the role of the field where the user can write, start programs or load pictures. The working area is scrollable; one can scroll it up or down by clicking on the scrolling points and dragging it in a scroll direction. It works in a similar way as the scrolling of the sliding blackboard.

Only a toolbox, placed by default on the right side of the working area, is reserved for special actions that are required during every E-Chalk lecture. Thus, the working area resembles the traditional blackboard, while the digital

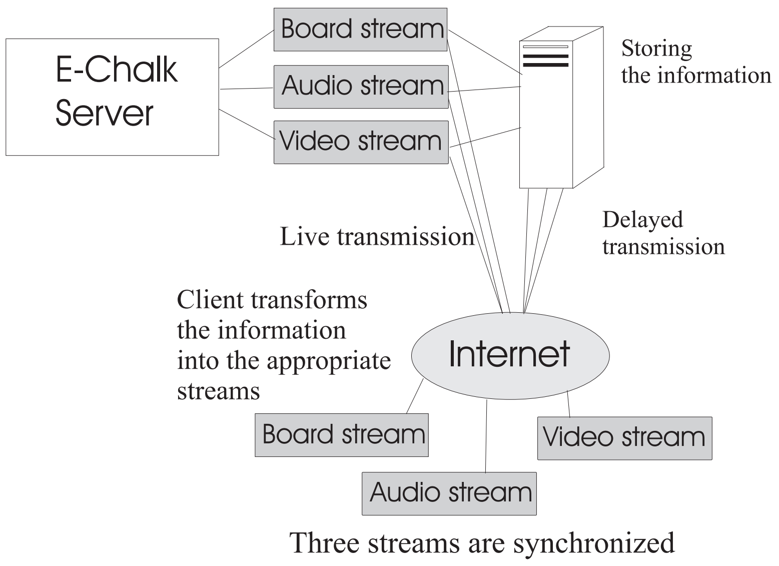


Figure 9.3: The architecture of the E-Chalk system.

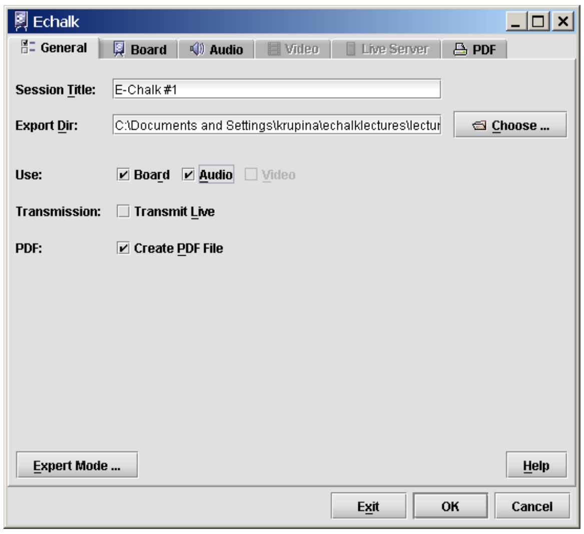
properties of the lecture are accessible within the toolbox. One can change the color and thickness of the pen, as well as undo, redo and erase what has been written on the board. There are also buttons on the toolbox for activation of the multimedia elements.

The E-Chalk team tried to avoid using a mouse and a keyboard corresponding to the main idea of the E-Chalk system. As in the traditional case, the lecturer is standing near the board rather than sitting in front of a computer. Therefore, algorithms for handwriting recognition were developed and integrated into E-Chalk system. [19, 18, 68].

9.1.4 Intelligent assistants in E-Chalk

Specialized applications called "intelligent assistants" are the kind of multimedia elements that can be embedded in the E-Chalk lecture [24]. Intelligent assistants can be implemented by using an open API, which was developed by Lars Knipping [39] as a part of his PhD studies. The idea of the intelligent assistants is that they can generate "intelligent" responses on certain actions from the lecturer on the board, e.g. drawn elements or written commands. Therefore, the context of the E-Chalk board can be extended with new information, generated on demand by these applications. That gives the flexibility to illustrate the lecture with graphs or animations prepared in real time.

As an example of an intelligent assistant, the well-known game "tic-tac-toe" was realized. As one can see in Fig. 9.6, the game field is defined as a part of the working area when starting the "tic-tac-toe" application. The user can



Session Title: E-Chalk #1

Export Dir: C:\Documents and Settings\krupina\echalklectures\lectur

Use: [✓ Board] [✓ Audio] Video

Transmission: Transmit Live

PDF: [✓ Create PDF File]

Figure 9.4: The settings panel in the E-Chalk Startup Wizard.

draw the symbol "O" and the computer provide "intelligent" response acting as a competitor with the symbol "X".

The intelligent assistants can, like other kinds of multimedia elements, be activated using a reserved button on the toolbox. All archived versions of such programs are added in the list of bookmarks used in the E-Chalk Startup Wizard. After activating the button corresponding to the intelligent assistants, the user can select the program from the list. He must also specify the area reserved for the application. Within the application area all strokes will be controlled by the application generating responses. The application will be stopped when the user starts to work outside of the "application area".

The main class of the E-Chalk API is called "Chalklet". Since NeuroSim was integrated into E-Chalk using "Chalklet" as the main class, I will review the main methods of the "Chalklet" interface. A brief review of other classes of the E-Chalk API will also be given.

9.2 API of the intelligent assistants

The main class for interactive board animations, **Chalklet**, is derived from the class **StrokeListener**. **StrokeListener** "listens" for any stokes drawn by the user in the animation area. One of the most important methods of **Chalklet** for development of an intelligent assistant is

- *pushStroke(BoardStroke aBoardStroke)*, which is called when a stroke is drawn in the animation area. The new stroke will be inserted into the

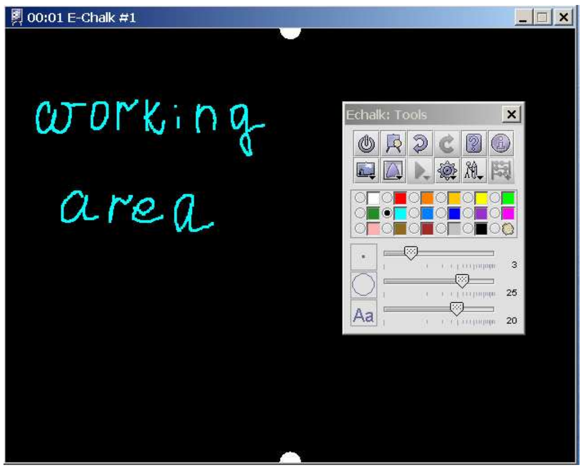


Figure 9.5: The working area and the toolbox as they appear during an E-Chalk session.

buffer. The developer can define the actions that should be taken in response to such a stroke.

The next important class in the E-Chalk API is **ChalkletContext**, which opens a connection between **Chalklet** and the board. An important step in the development of an intelligent assistant is overwriting following methods, used for strokes to be painted in the animation area:

- *sendStroke(BoardStroke aBoardStroke)* requests a stroke that is created by the Chalklet itself in order to be displayed
- *sendStrokes(BoardStroke[] anArrayOfABoardStroke)* performs the same as sendStroke, but for an array of strokes.

The class **ChalkletKit** is a factory class for chalklets. The method *getChalklet()* returns an instance of the class Chalklet. An instance of a **ChalkletKit** element is created by calling the constructor with the parameters of the application. For example, in the NeuroSim intelligent assistant one uses the name of the simulated model as a parameter.

These parameters are used by the animation invocation and are defined by the class **KitSetupInfo**. This class contains inner classes for different variable types. The values of the parameters can be specified with the help of the E-Chalk Startup Wizard.

As mentioned above, an important step in the development of "intelligent assistants" is overriding the method *pushStroke* of the class Chalklet and invoking *sendStroke* from ChalkletContext. The method *pushStroke* can analyze strokes drawn by the user inside the animation area using methods of the class BoardStroke. By default, the method *pushStroke* only inserts a new stroke into

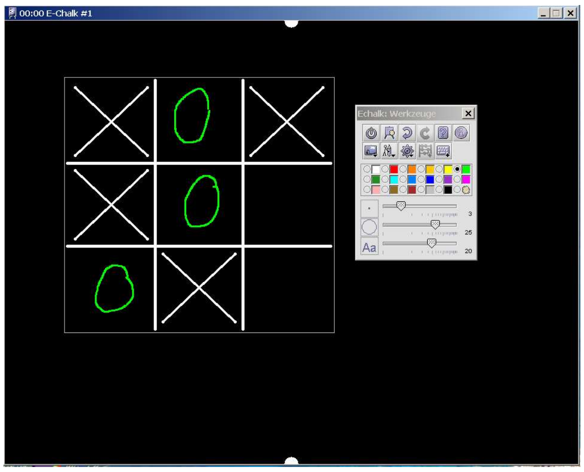


Figure 9.6: “Tic-tac-toe” animation embedded in E-Chalk.

the buffer. But it can also take certain actions, according to certain forms of the analyzed stroke as defined by the overriding. After overriding, it defines how the application should react to the drawing context in the animation area and possibly sends some strokes to the board.

9.3 Handwriting recognition in NeuroSim

By developing NeuroSim as an integrated part of E-Chalk, such GUI elements, for model definition and properties setting, as menu, pop-ups or windows were substituted by an interface in the form of a conventional blackboard. Handwriting recognition was used in the conventional way for a board environment. The user defines a model, sets the properties of simulated elements, writes on freehand the commands for performing a simulation and the system recognizes all that has been drawn and takes necessary actions.

For this purpose, we integrated the handwriting recognition system developed by Ernesto Tapia [19] in NeuroSim. It is based on the principle of online recognition, which means that it uses the dynamic information of written symbols (coordinates and temporal information). The recognition consists of three parts: preprocessing, classification and postprocessing. During the preprocessing the system prepares the information relevant for the classifier. Strokes are smoothed, scaled, ordered, and grouped into symbols. Thus, the input sequences of strokes are transformed into feature vectors using the preprocessing algorithm proposed in [21]. As features the coordinates of a point, the length of the line stroke, its relative length and the center of gravity can be considered.

The feature vectors will be used by the classification system, which is implemented in the form of vector support machines [8]. The system was previously trained to recognize the definite set of symbols required for NeuroSim.

The last step of recognition is postprocessing. In this step, symbols may be reclassified according to the context information. For example, the symbols “l” and “e” could be written in the same way and could then only be distinguished by analyzing the context.

After recognition, the program should perform some actions, e.g. setting properties or starting an animation. Strokes drawn in the application area will be processed by the intelligent assistant.

The main class of the system of handwriting recognition is derived from the class **StrokeListener**. The system recognizes all drawn strokes if some rule is fulfilled (if one writes with the color reserved for recognition).

The method *pushStroke* of this class is invoked to pass on strokes for recognition. The handwriting recognition is performed when one of the simulated neural elements is encircled with a certain color. The board then displays the properties of this element. After that, the user may write the expression in the form “name_of_the_parameter = value” which should be recognized. When the symbol `\end` is entered in the last position, the classifier will start recognition of the expression (see Fig. 9.7). If this process is completed successfully, the strokes are erased from the board and the value of the parameter is set.

The method *popupResult* of the class HandwritingRecognition returns the result of the string recognition. A string is divided by the token “=” into two substrings: the name of the symbol and its value. One restriction is imposed on the recognized string content: only letters are expected in the first part of the string and only numbers (the value of the parameter) in the second part. This operation is performed during the postprocessing step of the recognition.

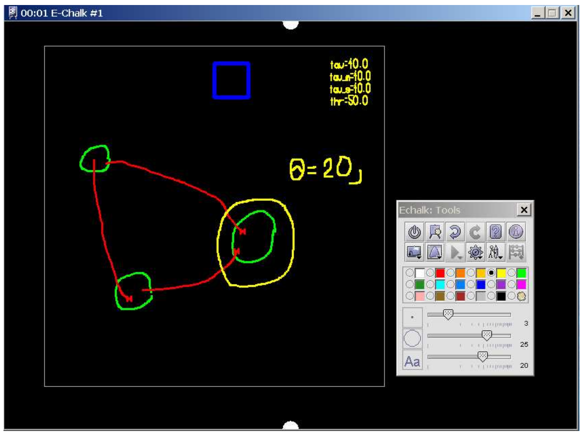


Figure 9.7: The parameters of neural elements can be set using handwriting recognition.

9.4 Integration of NeuroSim into E-Chalk

Let us now consider the features of the NeuroSim simulation system as integrated into an electronic chalkboard.

The lecturer performs the whole process of modelling by drawing and writing on the board, so that the natural environment of the blackboard is preserved. The drawn model elements, their connections and simulation tasks are recognized by the application. Properties of neurons, channels, and connections are defined directly by handwriting on the board. Control of the simulation is also performed by handwritten commands.

Different colors are reserved for different operations. Yellow is used for handwriting recognition. When the user draws in green, the system knows that a neuron should be added to the model. Red is used for inhibitory connection establishment, blue for excitatory connections.

Let us now look how the neural model can be designed with E-Chalk. When the user draws a neuron on the board in the form of a closed stroke, an element of type Neuron is created and added to the model. Following that, the connections may be established by drawing a stroke from one neuron to another. The start and end points of this stroke should belong to the bounding box of a stroke representing neurons, which will be connected. The additional field of the BoardStroke type in the classes of neurons and their connections is implemented to describe a visual representation of the element on the chalkboard.

Next, the parameters of the simulation should be set, drawing their names and values by hand. These will be recognized by the handwriting recognition module and passed on to the model.

The defined model can be edited by deleting and adding the elements of which it is composed. Deleting neurons and connections can be done with the erasing tool from the E-Chalk menu. Deletion is performed by drawing a stroke in the background color across the elements. The strokes representing a neuron or connection on the board that are touched by that erasing motion will be erased from the screen. Accordingly, the neuron or connection will be removed from the model description.

A command area reserved for writing commands is marked with a rectangle (see Fig. 9.8). It was decided to use capital letters as an abbreviation of the full command names. When written inside this “box”, they are recognized as commands for executing the simulation as well as opening or saving files with model descriptions. The result of the recognition process is shown in typed form after repainting the expression to indicate command recognition.

To give the possibility of presenting information in the form of typed strings, I have developed a class that contains functions for generating a set of strokes used for symbol generation. Typed symbols are represented as a set of straight lines and arcs. Strokes are scaled according to the size of the generated letters.

Fig. 9.8 illustrates working in the command area by example of the command to execute a simulation: the letter “R” (Run), as it was written by hand and after recognition. The program confirms recognition by generating in the typed form of the letter, and then starts the simulation.

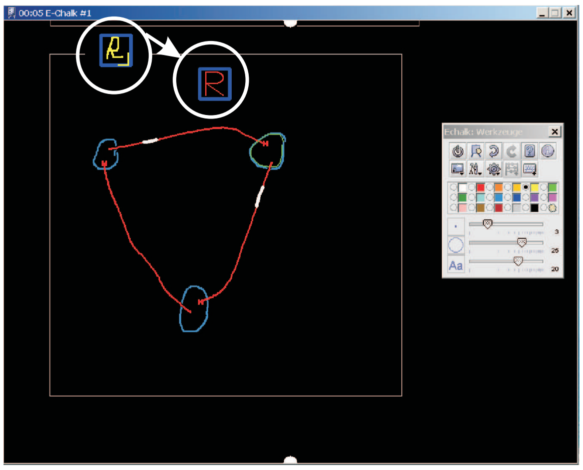


Figure 9.8: Executing a simulation of pulsed neural networks integrated into E-Chalk.

Models so created can be saved in an XML file to be used in the next session. Neurons and their connections are saved together with their properties. If a neural network was simulated, the dynamics of neural processes were held in queues of neural elements and all dynamic information can be saved in separate files. In addition, files containing data describing strokes of neurons and connections are also saved. This information is necessary to reconstruct the model as it was previously defined on the E-Chalk board. Thus, model descriptions are saved in linked files of three different types (see Fig. 9.9). To load a model, all of these files for each neural element need to be loaded.

When the user writes by hand the symbols "S"(Save) or "O" (Open) inside the area reserved for writing commands, the simulated models are saved to disk or files describing a previously simulated model are opened. Upon opening, the information is loaded from the disk and the model is constructed. Neural elements will be drawn according to previously saved graphical information.

9.4.1 Different neural model types

It is suggested to use the type of neural model as the parameter for the Neuron-Chalklet. The simulations of two models will be shown: pulsed neural networks and real neurons composed of voltage-gated channels with Hodgkin-Huxley dynamics. The simulation of pulsed neurons is performed online, i.e. the calculations and results presenting are performed on the local computer running E-Chalk. The Hodgkin-Huxley neuron simulation is realized with a client-server architecture, like the one on which NeuroSim is based.

The model type is defined, when one adds the animation to the list of the "intelligent assistants" using the E-Chalk Startup Wizard. Figure 9.10 shows

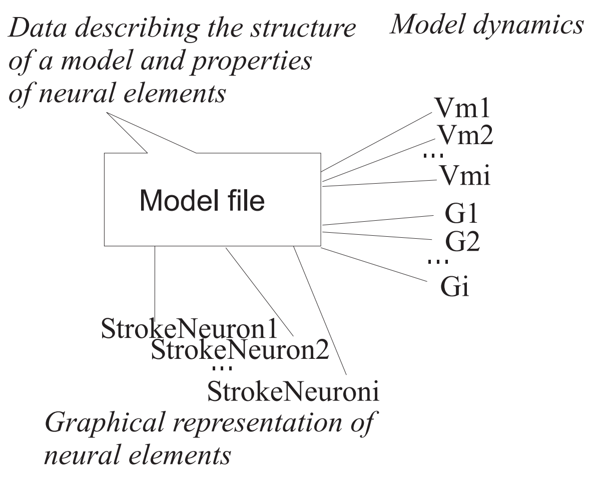
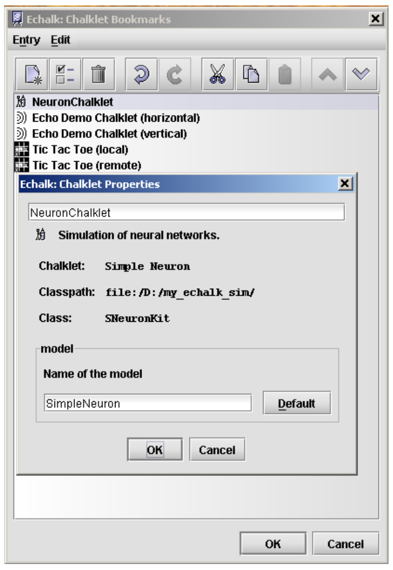

Data describing the structure of a model and properties of neural elements
Model dynamics
Model file
Vm1
Vm2
...
Vmi
G1
G2
...
Gi
StrokeNeuron1
StrokeNeuron2
...
StrokeNeuroni
Graphical representation of neural elements

Figure 9.9: Structure of the description files of neurons, simulated with E-Chalk.

a menu of parameters for the NeuronChalklet in an E-Chalk Startup Wizard window.


Echalk: Chalklet Bookmarks
Entry Edit
NeuronChalklet
Echo Demo Chalklet (horizontal)
Echo Demo Chalklet (vertical)
Tic Tac Toe (local)
Tic Tac Toe (remote)
Echalk: Chalklet Properties
OK
Cancel
Default
OK
Cancel

NeuronChalklet

Simulation of neural networks.

Chalklet: Simple Neuron

Classpath: file:/D:/my_echalk_sim/

Class: SNeuronKit

model

Name of the model

SimpleNeuron

Figure 9.10: Parameter assignment for a simulated neural model in the E-Chalk Startup Wizard.

9.4.2 Simulation of pulsed neurons

The *Spike Response Model* was taken as an example of simulation of pulsed neural networks because it is a a well-known basic model of pulsed neural networks (see Section 2.2.3). The screenshots from the animation of a simulation, which are shown in Fig. 9.11, represent the active neural network as defined in E-Chalk. The visual presentation of the model elements in E-Chalk during the animation is kept in the form as it was drawn upon model definition. When the neuron is active, it will be redrawn in blue. The level of a neuron's activity is shown by filling the inner area of the neuron with the strokes resembling the form of the neuron.

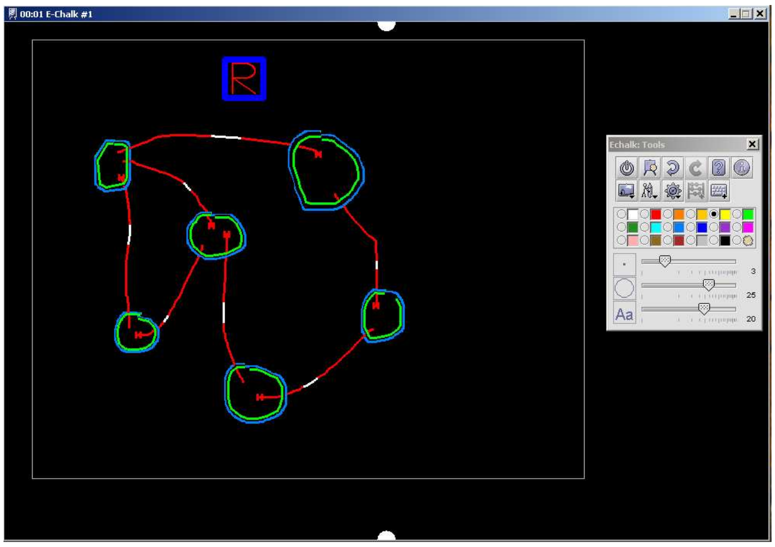


Figure 9.11: An animation of a pulsed neural network in E-Chalk .

Interactions between pre- and postsynaptic neurons are displayed by short pulses transferred via connections between the neurons. When the activity of a presynaptic neuron reaches the threshold to fire, it will influence the activity of the postsynaptic neuron. The contribution from presynaptic neurons is taken into account, as well as the connection weight and the transmission delay.

The following properties are the most important for the *Spike Response Model*:

- the time constant τ is included in a refractory function describing the contribution from presynaptic neurons
- the time constants τ_m and τ_s are included into the function describing the contribution of the presynaptic spike's response
- a threshold θ defines the level of neural activation

The values of these properties can be observed on the board and edited using handwriting recognition (see Fig. 9.7).

9.4.3 Simulation of real neurons with Hodgkin-Huxley dynamics

As the second example, I chosen a model of real neurons composed of voltage-gated channels with Hodgkin-Huxley dynamics. Each neuron consists of two compartments corresponding to a soma and dendrite. A soma contains sodium (Na^+) and potassium (K^+) Hodgkin-Huxley channels. A dendrite contains synaptically activated channels. Neural connections of two types can be established: excitatory and inhibitory. An excitatory connection is established by a sodium synaptic channel; inhibitory connections correspond to potassium synaptic channels.

As in the previous example, neurons can be defined by drawing closed curves in green. Stimulated injection currents are marked with a white "arrow" stroke. Excitatory connections are drawn in red; inhibitory connections are marked in blue.

Numerical integration is performed on the server. Following that, the animation of neuron activations and propagation of pulses can be played back. Writing free-hand the symbol "C" (Calculate) inside the "commands area" gives the command to begin the simulation. Then, the model description data is transferred to the NeuroSim server for calculation and the results will be sent back to the client. The command "Run" ("R") allows playback of the data received from the server. As with NeuroSim, the activation of a neuron is indicated with an activation color and the propagation of pulses through the neural connections is displayed with running strokes. The dynamics of the simulation can be analyzed with the membrane potential curves for different compartments. One can see in Fig. 9.12 that the animation area is divided into two parts: the area where the model is presented and the area where neuron activation curves are displayed. Figure 9.12 displays the animation process of a four-neuron network.

Fig. 9.13 shows the editing procedure for the property values of neurons. One chooses the neuron by drawing a closed stroke around it; then its properties become visible on the board and the properties editing mode is activated. The properties of the two compartments of which each neuron is composed are represented with a distinguishing symbol before the property's name: for soma properties we use the prefix "s_", for a dendrite "d_". Figure 9.13 shows that the user has written on the board the stroke "$I = 0$"; it sets the value of the injection current applied to the soma to zero. The properties editing mode can be closed by drawing a line through the area where the properties and their values are shown (see inset of Fig. 9.13). The drawing strokes are then be erased from the board.

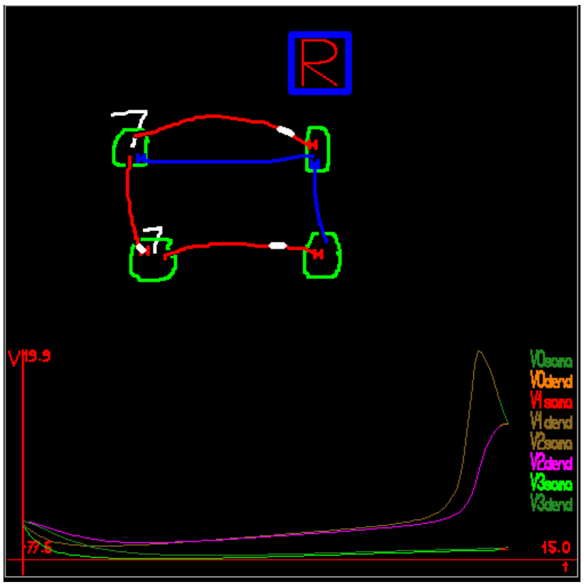
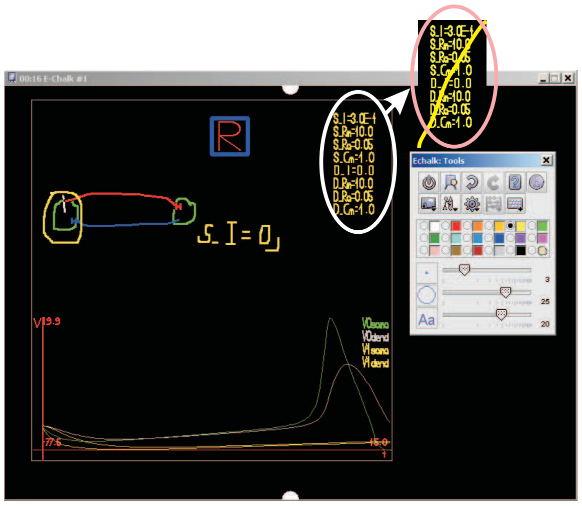


Figure 9.12: Model of a real neural network, simulated with E-Chalk.



S_I=3.0E-4
S_Rm=10.0
S_Ra=0.05
S_Cm=1.0
D_I=0.0
D_Rm=10.0
D_Ra=0.05
D_Cm=1.0

Figure 9.13: Properties editing mode for a Hodgkin-Huxley neuron.

9.5 Summary

To summarize this chapter, I propose that we have opened up new possibilities for giving neuroscience lectures. Our program, integrated in the electronic chalkboard E-Chalk, can be effectively used for educational purposes.

The basic features of the E-Chalk system have been introduced. The main concept in its development was to improve the functionalities of traditional classroom teaching, combining them with all the capabilities of a digital system.

The main concept of E-Chalk that we used is the possibility for the user to develop and integrate special-purpose programs. Such programs can work in the background, emphasizing certain aspects of the lecture material. The main functions of the E-Chalk API were reviewed.

In the next section, the system of handwriting recognition was described. It is used in my program for assigning properties of neural elements and entering commands in a running simulation. Thus, using the keyboard and mouse is not necessary during use of my program, so they do not interfere with the interface of the board.

The principles of working with NeuroSim as an integrated E-Chalk assistant have been shown in two examples. Simulations of pulsed neurons, in contrast to simulation of real neural networks, can be started locally. Simulations with E-Chalk of real neural networks, based on compartmental modelling, have been realized as an application with a client-server architecture.