MIMIC — A Language for Specifying Facial Animations

Thomas Fuchs, Jörg Haber, Hans-Peter Seidel

Max-Planck-Institut für Informatik
Stuhlsatzenhausweg 85
66123 Saarbrücken, Germany
`{tfuchs,haberj,hpseidel}@mpi-sb.mpg.de`

ABSTRACT

This paper introduces a versatile language for specifying facial animations. The language Mimic can be used together with any facial animation system that employs animation parameters varying over time to control the animation. In addition to the automatic alignment of individual actions, the user can fine-tune the temporal alignment of actions relatively to each other. A set of pre-defined functions can be used to control oscillatory behavior of actions. Temporal constraints are resolved automatically by the Mimic compiler. We describe the grammar of Mimic, give some hints on the implementation of the Mimic compiler, and show some examples of animation code together with snapshots from the resulting animation.

Keywords: facial animation, script language, time-dependent animation parameters

1 INTRODUCTION

For interpersonal communication and social interaction, the human face is the most important part to convey emotions and psychological states. In the past decades, many efforts have been made to develop a variety of facial modeling and animation techniques. Today, facial animations are used in computer games, movies, advertising, and education. Almost every computer game published in the last few years features animated characters, many of them representing humans. More and more movies include virtual human actors to show sequences a real human would not be able to perform, and many commercials make use of animated humans to disseminate their messages. Educational software frequently includes virtual teachers to demonstrate tasks or to help the user through the program.

The quality of facial animations in several recent movies is very high. It is sometimes hard to tell virtual sequences from real ones. To achieve this level of quality, however, a huge amount of manual interaction is typically required to create the animations. Most of the research in facial modeling and animation has striven to enhance the visual quality of models and animations. Unfortunately, the problem of *how to establish optimal control functionality* for the creation of facial animations has obtained comparatively little interest in research.

In this paper, we present an abstract and thus versatile language for specifying facial animations. The language Mimic does not depend on any particular animation system. Mimic can be used together with any animation system that employs animation parameters varying over time to control the animation. Such animation parameters are, for instance, the MPEG-4 FAP's [ISO00], the action units (AU) from the FACS system [EF78], or (pseudo-)muscle contraction values. The language Mimic allows actions to take place sequentially and/or in parallel. Additional directives to control the temporal alignment of actions relatively to each other are available as well as a set of pre-defined functions that are used to control oscillatory behavior of actions. All temporal constraints are resolved automatically by the Mimic compiler. The resulting actions are discretized in time, taking into account the temporal variance of each animation param-

WSCG SHORT Communication papers proceedings
WSCG'2004, February 2–6, 2004, Plzen, Czech Republic.
Copyright UNION Agency — Science Press

eter. Finally, the values of animation parameters that are influenced by several simultaneous actions are blended in one of several different ways, which can be specified by the user. For each animation parameter, a sequence of time/value pairs is generated, which can be read from the user's preferred animation system.

2 PREVIOUS WORK

During the last three decades, many different approaches to facial animation haven been proposed. The book by Parke and Waters [PW96] provides a comprehensive though not very recent overview of the topic. In general, the techniques proposed can be classified into parametric models [Par74, Par82, Wat87, MTPT88], physics-based approaches [PB81, TW90, LTW93, LTW95, KHS01], expression-blending methods [PHL$^+$98, BV99, NN01, LSZ01], and performance-based techniques [Wil90, GGW$^+$98].

To generate an animation, both parametric models and physics-based approaches require some (in fact, usually many) parameters to be specified and modified over time. These parameters are, for instance, the *Facial Animation Parameters* (FAPs) from the MPEG-4 standard [ISO00] or (pseudo-)muscle contraction values as used in physics-based approaches [LTW95, KHS01]. Specifying all of these parameters manually can be a tedious and awkward piece of work, which sometimes even takes longer than building the face model itself. Thus, several approaches have been proposed to automatically generate facial animations from a high-level user description. These control approaches can be categorized based on different criteria. Pelachaud *et al.* [PBV94] concentrate on the underlying philosophy of the approaches and distinguish rule-based, analysis-based, and performance-based approaches.

Rule-based approaches employ rules concerning the links between intonation, emotion, and facial expressions. These rules have been obtained from psychological and linguistic studies, see [Pel91] for an overview. Due to the link between speech and facial expressions, most of the rule-based approaches either include a text-to-speech component [PWWH86, HPW88, IC96, PBS96, AHK$^+$02] or at least support synchronization to an external speech system [KMMTT91]. Other rule-based methods use recorded natural speech signals to control the animation [PBS91, AHS02].

Analysis-based approaches [EP93, Bra99, EGP02] analyze video sequences of human faces (with or without additional audio) and store the results of the analysis in some internal data base. From this data base, arbitrary sequences of facial animation parameters are generated that result in naturally looking facial animations. The user only has to provide a control signal (usually a speech signal) for the desired animation, which is then composed automatically from the information stored in the data base.

Performance-based approaches are tightly coupled to performance-based animation systems [Wil90, GGW$^+$98]. Actually, the terminology is somewhat delusive: a parametric animation system that is controlled by a performance-based approach would probably be referred to as a performance-based animation system. Thus, performance-based control approaches can be seen as an ingredient that turns any kind of animation system into a performance-based animation system.

In this paper, we present a script-based approach for facial animation. In contrast to purely rule-based approaches, script-based systems are more general and flexible. In particular, synchronization of independent facial actions can be controlled easily up to the desired level of granularity. Unlike analysis-based methods, our approach does not require complex computations and data processing. Furthermore, we do not need any sophisticated hardware such as it is required for performance-based approaches.

3 LANGUAGE DESIGN

The design of Mimic is based on a small set of clear and intuitive principles, which are explained in the following subsections. A basic observation in many scientific areas is that complex processes can be decomposed in several subprocesses with less complexity. Recursively repeating this decomposition for every subprocess, one ends up with a number of indivisible subprocesses. In this paper, a process is called an *action* and an indivisible subprocess is denoted as an *atomic action.*

3.1 Sequences and Parallels

The human mind processes events in categories like "one after another" and "at the same time". The same categories are used to describe complex processes, resulting in terms such as "*first... then...*" or "*after that*" for serial events and "*while*" or "*during*" for simultaneous events. This concept of serial/parallel events proves to be both intuitive and adequate to describe complex animation sequences.

For individual actions, there are two types of temporal relationships between them: actions that happen one after another are called a *sequence*, while actions taking place simultaneously are called a *parallel*. In MIMIC, both sequences and parallels are actions themselves and can be nested recursively.

From a semantics' point of view, the main difference between sequences and parallels is the determination of the starting point of each action. For a sequence, the first action starts at the beginning of the sequence and every other action starts immediately after its preceding action ends. For a parallel, all actions start at the same time, i.e. at the beginning of the parallel.

3.2 Temporal Adjustment

In general, the starting point of an action is completely determined by its context. It is desirable, however, to allow the user to modify the default starting point of an action in order to increase the flexibility of the language. Thus, actions in sequences as well as in parallels can be shifted backward and forward with respect to their default starting point. Within a parallel, the ending point of an action can be aligned relatively to the end of the parallel.

3.3 Macros

While coding an animation, it often happens that the same set of actions is needed more than once. It would then be nice to specify the actions only once and use multiple instances of this definition throughout the animation. Such functionality can easily be provided through macros, similar to a "`#define ...`" in the C / C++ language.

4 LANGUAGE DEFINITION

The grammar of MIMIC is described in Table 1 in Backus-Naur-Form (BNF). In this table, the keyword `IDENTIFIER` is used to represent any parameter of the underlying facial animation system, for instance the opening angle of the jaw or a (pseudo-)muscle contraction value (see also the example in Table 2).

A MIMIC-program consists of two parts: a definition section and the main program. In the definition section, macros for complex actions and/or facial expressions can be defined using the def and defxpr keywords. An example using a macro for a smiling expression is shown in Figure 3.

The main program is an action group. Action groups are the basic building blocks of the MIMIC language. Every action group is either a sequence enclosed in curly braces {} or a parallel enclosed in brackets `[]`. Both sequences and parallels contain actions, which are either action groups or atomic actions. Each atomic action controls the temporal behavior of an animation parameter. This is done by either explicitly specifying a list of times and associated parameter values, a periodic function, or a fade-in-fade-out statement. A periodic function generates a list of values changing periodically over time. The fade-in-fade-out statement produces a list of values reaching a certain value, holding it for a specified amount of time, and fading out again.

Atomic actions can be controlled using a set of parameters. There are three types of parameters:

- parameters with assigned values: in, hold, out, value, duration, freq, period, min, max;
- parameters with optionally assigned values: left, right;
- parameters without values: linear, sin, log, exp, connect, average, add, overwrite.

The alignment of an action is controlled by the left and right parameters with an optional value indicating how much padding should be inserted. Padding values can be positive or negative, corresponding to a shift along the positive or negative time axis, respectively. Omitting these parameters is equivalent to specifying the default left=0. Right alignment is only supported in parallels, where the reference point for the alignment is given by the ending point of the longest action within the parallel.

The parameters in, out, hold, and value specify the duration of fade-in / fade-out and how long to retain the animation parameter value in-between. Fading in and out can be carried out using one of several different fading functions: linear, sin, log, and exp. For instance, the action "*jaw* <in=2,hold=3,out=1,value=0.3,log>" specifies the animation parameter *jaw* to fade-in logarithmically for two seconds, hold the value of 0.3 for three seconds, and fade-out for one second. The default behavior for fading out is to return to the animation parameter value that was present before fading in. However, it is sometimes desirable to fade out taking into account the next value of the same animation parameter that will follow in time. This is achieved by specifying the optional connect parameter: the target value for the fade-out is set to be the next value of the same animation parameter in time.

program	::=	def_list MAIN < parameter_list > actiongroup
	|	def_list MAIN actiongroup
def_list	::=	ϵ
	|	def_list def
def	::=	DEF `IDENTIFIER` actiongroup
	|	DEFXPR `IDENTIFIER` `FNAME`
actiongroup	::=	sequence
	|	parallel
sequence	::=	{ action_list }
parallel	::=	`[` action_list `]`
action_list	::=	action
	|	action_list action
action	::=	`IDENTIFIER` < parameter_list > (timevalue_list)
	|	`IDENTIFIER` (timevalue_list)
	|	`IDENTIFIER` < parameter_list > ;
	|	`IDENTIFIER` ;
	|	< parameter_list > actiongroup
	|	actiongroup
parameter_list	::=	parameter
	|	parameter_list , parameter
parameter	::=	pname
	|	pname_arg = `DECIMAL`
	|	pname_arg = `IDENTIFIER`
pname	::=	LEFT | RIGHT | LINEAR | SIN | LOG | EXP
	|	AVERAGE | ADD | OVERWRITE
pname_arg	::=	LEFT | RIGHT | IN | HOLD | OUT | VALUE | CONNECT
	|	DURATION | FREQ | PERIOD | MIN | MAX
timevalue_list	::=	time_list ; value_list ;
time_list	::=	ϵ
	|	time_list `DECIMAL`
value_list	::=	ϵ
	|	value_list `DECIMAL`

Table 1: Grammar of MIMIC in BNF. An ϵ represents the empty word and indicates the termination of a recursive rule. Parameters of the underlying facial animation system are marked by the keyword `IDENTIFIER`, while `FNAME` denotes a file name. `DECIMAL` represents a decimal number. See Section 4 for more details.

Animation parameter values changing periodically over time can be generated using the duration, freq, period, min, and max parameters. An additionally specified function (usually sin or linear) oscillates for a given duration between the min and max values with a user-defined frequency (or a user-defined period). In a parallel, the duration parameter can be omitted, in which case the oscillation spreads over the whole length of the parallel. In the example "*jaw* <sin,min=0.1,max=0.3,freq=2,duration=5>", the animation parameter *jaw* oscillates in a sinusoidal way for five seconds between the values 0.1 and 0.3 with a frequency of 2 Hz.

Finally, the blending parameters average, add, and overwrite are used to control the way in which multiple occurrences of a single animation parameter at a given time are combined. Such a situation can happen if, for instance, a complete facial expression (e.g. a smile) defined by the defxpr macro is overlaid in time by an additional action (e.g. lowered mouth corners), which modifies an animation parameter that is also present in the predefined facial expression. In this example, the mouth corners should move upwards due to the smile and downwards due to the explicit action. Depending on the setting of the blending parameter, the concurrent animation parameter values are either averaged, added up, or the parameter values from the action marked with overwrite simply overwrite the other ones. The default behavior is to add up animation parameters.

5 IMPLEMENTATION

MIMIC is implemented as a compiler, which reads a MIMIC-program and outputs animation parameter values. The compiler operates in four stages: scanning, parsing, semantical analysis, and code generation. Scanner and parser are implemented using the standard UNIX tools `lex` and `yacc` (or their public-domain equivalents `flex` and `bison`).

```
 1  main {
 2  [
 3    eyelid_right <min=0.325,max=0.375,period=5,sin>;
 4    {
 5      jaw <in=5,hold=15,exp,out=10,value=0.1,connect>;
 6      jaw ( 0    1     2    3     4     5;
 7            0.2  0.25  0.2  0.27  0.19  0.22; )
 8      [
 9        eyelid_right ( 0    37;
10                       0.1  0.2; )
11        jaw <left=15,min=0.1,max=0.2,period=5,
12             duration=10,linear>;
13        jaw <left=10,in=5,hold=10,sin,out=5,value=0.2>;
14      ]
15      jaw <in=5,hold=10,exp,out=10,value=0.07,connect>;
16      [
17        jaw <in=10,hold=10,exp,out=5,value=0.1,connect>;
18        <left=10> {
19          jaw <min=0.1,max=0.15,period=3,
20               duration=10,linear>;
21          jaw <left=5,min=0.05,max=0.1,freq=0.4,
22               duration=10,sin>;
23        }
24      ]
25    }
26  ]
27  [
28    eyelid_right <min=0.345,max=0.365,freq=0.4,sin>;
29    jaw ( 0.1  5     20  ;
30          0.1  0.15  0.05 ; )
31  ]
32  [
33    eyelid_right <min=0.315,max=0.385,period=3,sin>;
34    {
35      jaw <in=5,hold=15,sin,out=10,value=0.1>;
36      jaw ( 0    1     2    3     4     5;
37            0.2  0.25  0.2  0.27  0.19  0.22; )
38      jaw <left=10,in=5,hold=5,linear,out=5,value=0.2>;
39      jaw <in=5,hold=10,exp,out=10,value=0.1,connect>;
40      jaw <in=10,hold=10,sin,out=5,value=0.3>;
41    }
42    eyelid_right <left=10> ( 0  20   40    70    90;
43                             0  0.1  0.05  0.15  0.0; )
44  ]
45  }
```

Table 2: A Mimic-program using the animation parameters *jaw* and *eyelid_right*.

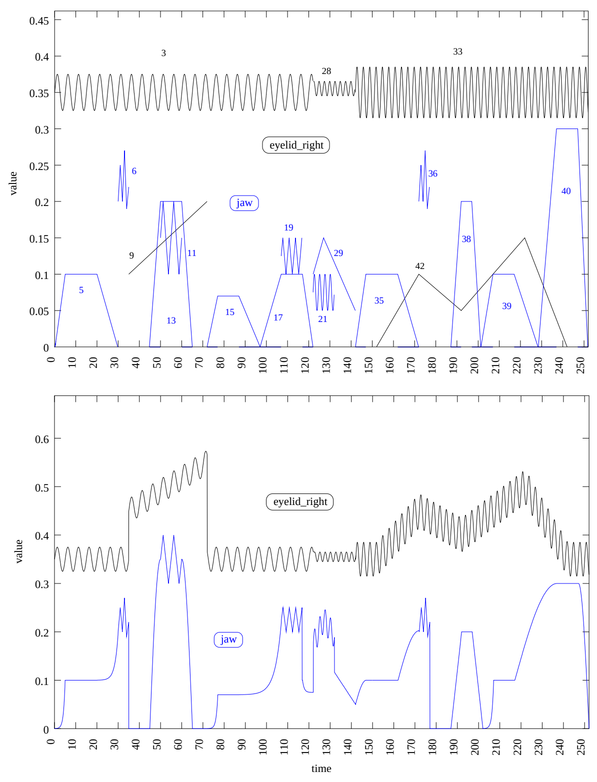


Figure 1: Temporal progression of the example from Table 2. The upper graph shows the progression of the parameter values for *jaw* and *eyelid_right* before combining and blending. The numbers correspond to the line numbers of the action in Table 2. The lower graph shows the final combined values with blending.

Some conditions and constraints that cannot be expressed in a context-free-grammar are checked during semantical analysis. These conditions are:

- every identifier that appears in a Mimic-program is either a valid animation parameter name or it has been defined in the definition section of the program;
- in a time-value list, the number of times must match the number of values;
- in sequences, right alignment of actions must not be used.

In the code generation step, a time-value list is generated for every animation parameter. The values are range-checked taking into account external constraints from the facial animation system. Finally, the time-value lists are thinned out if possible: parameter values that can be computed by linear interpolation from the neighboring values are removed together with their corresponding time stamp.

6 RESULTS

The full complexity of deeply nested sequences and parallels is hard to describe. Therefore we present and discuss two examples of animations written in Mimic in this section. The first example has been designed to explain the semantics of Mimic and to demonstrate the generation of animation parameters, while the second example shows some snapshots from the resulting animation sequence.

6.1 Example #1

Table 2 shows the Mimic-code of the first example. This animation consists of one sequence (line 1–45), which is composed of three parallels. The first parallel (line 2–26) consists of an atomic action (line 3) and a sequence (line 4–25). This sequence is composed of two atomic actions (line 5–7), one parallel with three atomic actions (line 8–14), another atomic action (line 15), and another parallel (line 16–24), which is composed of an atomic action (line 17) and a sequence (line

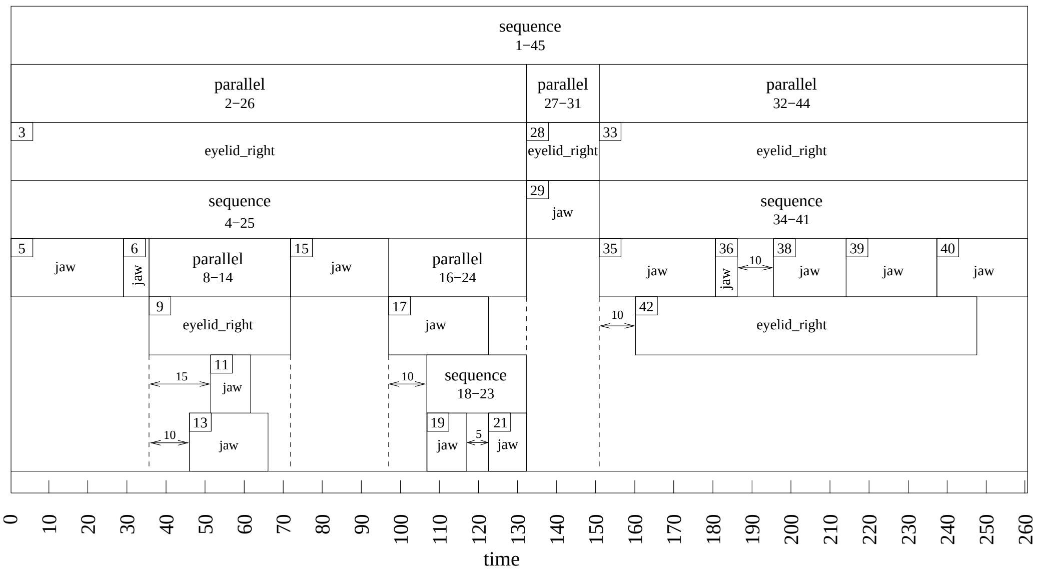


Figure 2: Visualization of the structure of the example shown in Table 2. See Section 6.1 for a detailed description.

18–23) delayed by ten seconds. The second parallel in the main sequence (line 27–31) consists of two atomic actions. Finally, the third parallel (line 32–44) is built from an atomic action (line 33), a sequence (line 34–41) consisting of five atomic actions, and another atomic action (line 42–43).

Figure 1 shows two plots of the temporal progression of the animation parameter values *jaw* (in blue) and *eyelid_right* (in black) resulting from the animation code listed in Table 2. The upper plot shows the values of the animation parameters as they are prescribed from each atomic action. These values are not yet combined into a single curve. The numbers indicate the corresponding line numbers from Table 2, where each action is defined. Actions that fade in and out are shown with linear flanks because the blending has not yet been calculated.

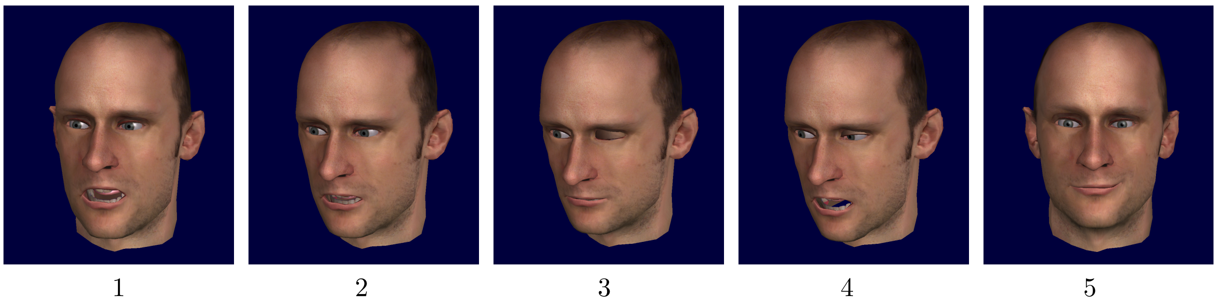
The lower plot in Figure 1 shows the final animation parameter curves for the parameters *jaw* and *eyelid_right*. The blending has been computed and the parameter values are combined into a single curve for each animation parameter. A simple blending example can be observed along the curve for the animation parameter *eyelid_right*. The final curve is obtained by combining the oscillating pattern (line 3) with the time-value list from line 9 and the sinusoidal pattern (line 33) with the time-value list from line 42 using the default blending method `add`.

The effect of the `connect` parameter is visible at the end of the *jaw* curve (corresponding to lines 38–40 in the code). The action from line 38 fades in and out linearly. The target value for fading out is identical to the value of the animation parameter before fading in, which is zero in this example. In lines 39 and 40, the actions also fade in and out. However, the action from line 39 is connected to the action from line 40 by means of the `connect` parameter. As a result, the target value for fading out the action from line 39 is given by the animation parameter value from the next action (line 40).

Figure 2 illustrates the structure of the example. Every rectangle represents an action with the corresponding line number from Table 2 printed in the upper left corner. The placement and size of the rectangle correspond to the starting point and duration of the action. Sequences are depicted in a left-to-right order, parallels are arranged top-to-bottom. Horizontal arrows indicate indentations as defined by the `left` parameter with the duration printed above the arrow.

6.2 Example #2

Figure 3 shows another example together with a plot of the corresponding animation parameter curves and five snapshots taken from the resulting facial animation. This example is a sequence composed of a parallel and a pre-defined facial expression. The parallel consists of five atomic actions controlling the animation parameters *head_rot_y*, *eyelid_left*, *head_rot_x*, *jaw*, and *lookat_x*. The pre-defined expression represents a smile and is encoded in the file `"smile.xpr"`. The snapshots shown in Figure 3 have been generated using a physics-based facial animation system. Thus, muscle contraction values have been



```
defxpr smile "smile.xpr"
main {
[
  head_rot_y <in=5,hold=10,out=5,sin,value=-20>;
  eyelid_left <left=6,in=3,hold=2,out=1,sin,value=0.8>;
  head_rot_x <in=5,hold=10,out=5,exp,value=10>;
  jaw <min=0,max=0.09,freq=0.1,sin,duration=20>;
  lookat_x ( 0   12   20;
             0  100   0; )
]
smile <in=5,hold=10,out=5,value=1,sin>;
}
```

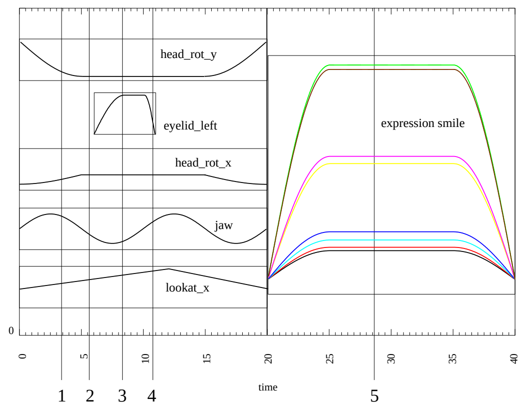


Figure 3: Snapshots from an animation sequence as generated by the MIMIC-program shown at the bottom left. The temporal progression of the animation parameters is shown in the graph at the bottom right. During the first twenty seconds, five different animation parameters proceed in parallel. During the next twenty seconds, a pre-defined facial expression (*smile*) is faded in, held, and faded out again. The colored curves on the right half of the graph represent individual muscle parameters that are defined in the file `"smile.xpr"`.

used to generate the smiling expression. The MIMIC compiler, however, does not know about the meaning of the animation parameters. Any set of animation parameters that generates a smile when input to the given facial animation system will work the same way.

7 CONCLUSION & FUTURE WORK

We have presented MIMIC, a versatile language for specifying facial animations. The language can be used together with any facial animation system that employs animation parameters varying over time to control the animation. The user specifies animations in terms of sequential and parallel processes, a concept which is intuitive and familiar to everybody. Additional fine-tuning of the temporal alignment of individual actions is supported as well as the use of periodic functions that control oscillatory behavior of actions. The MIMIC compiler automatically takes care of blending and thinning out animation parameter values.

In our experiments, we found that facial animations of arbitrary complexity can be specified easily using the MIMIC language. Coding the example from Table 2 took about ten minutes. Judging from the complex shape of the resulting animation parameter curves in Figure 1 (bottom), it probably would not have been possible to obtain the same result in the same time by directly specifying a time-value list for each animation parameter.

Obviously, there are still many ways for improving an animation language such as MIMIC. Coding an animation could be facilitated largely if the system knew about the dependencies between different animation parameters (e.g. movement of the eyelids while looking up and down) and would automatically include such dependent actions. In addition, a mechanism for speech synchronized animations should be introduced into MIMIC. To this end, the temporal alignment of actions could be extended to include the alignment of phonemes. Finally, the links between emotions, facial expressions, and speech are worthwhile to be investigated and included into our system.

REFERENCES

[AHK$^+$02] Irene Albrecht, Jörg Haber, Kolja Kähler, Marc Schröder, and Hans-Peter Seidel. "*May I talk to you?* :-)" — Facial Animation from Text. In *Proc. Pacific Graphics 2002*, pages 77–86, October 2002.

[AHS02] Irene Albrecht, Jörg Haber, and Hans-Peter Seidel. Automatic Generation of Non-Verbal Facial Expressions from Speech. In *Proc. Computer Graphics International 2002 (CGI 2002)*, pages 283–293, July 2002.

[Bra99] Matthew Brand. Voice Puppetry. In *Computer Graphics (SIGGRAPH '99 Conf. Proc.)*, pages 21–28, August 1999.

[BV99] Volker Blanz and Thomas Vetter. A Morphable Model for the Synthesis of 3D Faces. In *Computer Graphics (SIGGRAPH '99 Conf. Proc.)*, pages 187–194, August 1999.

[EF78] Paul Ekman and Wallace V. Friesen. *Facial Action Coding System. Manual.* Consulting Psychologists Press, Palo Alto, CA, 1978.

[EGP02] Tony Ezzat, Gadi Geiger, and Tomaso Poggio. Trainable Videorealistic Speech Animation. In *ACM Transactions on Graphics (SIGGRAPH 2002 Conf. Proc.)*, pages 388–398, July 2002.

[EP93] Irfan A. Essa and Alex Pentland. A Vision System for Observing and Extracting Facial Action Parameters. Technical Report #247, MIT Media Laboratory, Cambridge, MA, 1993.

[GGW$^+$98] Brian Guenter, Cindy Grimm, Daniel Wood, Henrique Malvar, and Frédéric Pighin. Making Faces. In *Computer Graphics (SIGGRAPH '98 Conf. Proc.)*, pages 55–66, July 1998.

[HPW88] David R. Hill, Andrew Pearce, and Brian Wyvill. Animating Speech: An Automated Approach using Speech Synthesised by Rules. *The Visual Computer*, 3(5):277–289, March 1988.

[IC96] Horace H. S. Ip and C. S. Chan. Script-Based Facial Gesture and Speech Animation Using a NURBS Based Face Model. *Computers & Graphics*, 20(6):881–891, November 1996.

[ISO00] ISO/IEC. Overview of the MPEG-4 Standard. `http://www.cselt.it/mpeg/standards/mpeg-4/mpeg-4.htm`, July 2000.

[KHS01] Kolja Kähler, Jörg Haber, and Hans-Peter Seidel. Geometry-based Muscle Modeling for Facial Animation. In *Proc. Graphics Interface 2001*, pages 37–46, June 2001.

[KMMTT91] Prem Kalra, Angelo Mangili, Nadia Magnenat-Thalmann, and Daniel Thalmann. SMILE: A Multilayered Facial Animation System. In *Proc. IFIP WG 5.10*, pages 189–198, Tokyo, Japan, 1991.

[LSZ01] Zicheng Liu, Ying Shan, and Zhenyou Zhang. Expressive Expression Mapping with Ratio Images. In *Computer Graphics (SIGGRAPH 2001 Conf. Proc.)*, pages 271–276, August 2001.

[LTW93] Yuencheng Lee, Demetri Terzopoulos, and Keith Waters. Constructing Physics-based Facial Models of Individuals. In *Proc. Graphics Interface '93*, pages 1–8, May 1993.

[LTW95] Yuencheng Lee, Demetri Terzopoulos, and Keith Waters. Realistic Modeling for Facial Animations. In *Computer Graphics (SIGGRAPH '95 Conf. Proc.)*, pages 55–62, August 1995.

[MTPT88] Nadia M. Magnenat-Thalmann, E Primeau, and Daniel Thalmann. Abstract Muscle Action Procedures for Human Face Animation. *The Visual Computer*, 3(5):290–297, March 1988.

[NN01] Jun-yong Noh and Ulrich Neumann. Expression Cloning. In *Computer Graphics (SIGGRAPH 2001 Conf. Proc.)*, pages 277–288, August 2001.

[Par74] Frederic I. Parke. *A Parametric Model for Human Faces.* PhD thesis, University of Utah, Salt Lake City, UT, December 1974.

[Par82] Frederic I. Parke. Parameterized Models for Facial Animation. *IEEE Computer Graphics and Applications*, 2(9):61–68, November 1982.

[PB81] Stephen M. Platt and Norman I. Badler. Animating Facial Expressions. In *Computer Graphics (SIGGRAPH '81 Conf. Proc.)*, volume 15, pages 245–252, August 1981.

[PBS91] Catherine Pelachaud, Norman Badler, and Mark Steedman. Linguistic Issues in Facial Animation. In *Computer Animation '91*, pages 15–30. 1991.

[PBS96] Catherine Pelachaud, Norman Badler, and Mark Steedman. Generating Facial Expressions for Speech. *Cognitive Science*, 20(1):1–46, 1996.

[PBV94] Catherine Pelachaud, Norman I. Badler, and Marie-Luce Viaud. Final Report to NSF of the Standards for Facial Animation Workshop, October 1994.

[Pel91] Catherine Pelachaud. *Communication and Coarticulation in Facial Animation.* PhD thesis, Unicersity of Pennsylvania, Philadelphia, 1991.

[PHL$^+$98] Frédéric Pighin, Jamie Hecker, Dani Lischinski, Richard Szeliski, and David H. Salesin. Synthesizing Realistic Facial Expressions from Photographs. In *Computer Graphics (SIGGRAPH '98 Conf. Proc.)*, pages 75–84, July 1998.

[PW96] Frederic I. Parke and Keith Waters, editors. *Computer Facial Animation.* A K Peters, Wellesley, MA, 1996.

[PWWH86] Andrew Pearce, Brian Wyvill, Geoff Wyvill, and David R. Hill. Speech and Expression: A Computer Solution to Face Animation. In *Proc. Graphics Interface '86*, pages 136–140, May 1986.

[TW90] Demetri Terzopoulos and Keith Waters. Physically-based Facial Modelling, Analysis, and Animation. *Journal of Visualization and Computer Animation*, 1(2):73–80, December 1990.

[Wat87] Keith Waters. A Muscle Model for Animating Three-Dimensional Facial Expression. In *Computer Graphics (SIGGRAPH '87 Conf. Proc.)*, volume 21, pages 17–24, July 1987.

[Wil90] Lance Williams. Performance-Driven Facial Animation. In *Computer Graphics (SIGGRAPH '90 Conf. Proc.)*, volume 24, pages 235–242, August 1990.