



NECA

NET ENVIRONMENT FOR EMBODIED EMOTIONAL
CONVERSATIONAL AGENTS

D9c: Assessment of Markup Languages for Avatars, Multimedia and Multimodal Systems

Hannes Pirker, Brigitte Krenn

ÖFAI



Version: 1.0

Date: 28 May 2002

Project ref. no.	<i>IST-2000-28580</i>
Project title	NECA: A Net Environment for Embodied Emotional Conversational Agents
Deliverable status	Public
Contractual date of delivery	28 Feb 2002
Actual date of delivery	28 May 2002
Deliverable number	D9c
Deliverable title	D9c: Assessment of markup languages for avatars, multimedia and multimodal systems
Type	Report
Status & version	Final
Number of pages	33
WP contributing to the deliverable	WP 9
Task responsible	ÖFAI
Author(s)	Hannes Pirker, Brigitte Krenn
EC Project Officer	Patrice Husson
Keywords	markup languages
Abstract (for dissemination)	description and assessment of markup languages with respect to their feasibility for NECA

Austrian Research Institute for Artificial Intelligence (ÖFAI)

Schottengasse 3

A-1010 Wien

Austria

CONTENTS

EXECUTIVE SUMMARY	5
PURPOSE	6
REQUIREMENTS FOR A MARKUP LANGUAGE IN NECA	6
MULTIMODAL MARKUP LANGUAGES	7
VHML	7
VHML Timing and synchronization	8
MPML	8
TVML	10
MARKUP FOR INDIVIDUAL REPRESENTATION LEVELS	12
Syntactic Markup	12
Part of Speech Tags in Microsoft SAPI	12
Syntactic Markup for Large Text Corpora	12
Speech Synthesis Markup	16
Speech Synthesis Markup Language (SSML)	16
Microsoft SAPI TTS XML	19
SML in VHML	19
Comparison of SSML, SML and SAPI	19
Facial Animation Specifications	23
FAML Facial Animation Markup Language	23
Facial Action Coding Scheme (FACS)	24
Minimal Perceptible Action MPA	24
MPEG -4 Facial Animation	24
Emotional Markup	28
MISCELLANEOUS MARKUP LANGUAGES FOR WEB-BASED APPLICATIONS	30
VRML Markup for 3D Objects and Spaces	30
SMIL	31
Media Object Modules	31
SMIL Timing	32
Player Technology	32
CONCLUSION	33
REFERENCES	34

Executive Summary

The purpose of NECA is to develop a platform for the implementation of emotional conversational agents for Web-based applications. Part of this task is the specification of a mark-up-language – called “Rich Representation Language (RRL)” – which is used mainly system-internally as interface between different components of NECA.

In this document existing markup-languages are assessed for their usability within NECA. The following types of markup-schemes are included in this survey:

- Markup-languages for the specification of avatars, including multiple modalities
- Markup-languages for the specification of certain aspects of avatars (e.g. emotion, speech)
- Multimedia standards

Purpose

The aim of this document is a description and assessment of existing markup languages for their feasibility for Neca. The focus is on the representation and temporal integration of verbal and nonverbal aspects of affective communication, such as scenario information, text, emotional disposition of the communicating characters, syntactic structure, phonological and prosodic information, facial expression, gesture and posture. In terms of existing markup languages we concentrate on multi-media/ multi-modal markup in general and the integration of speech synthesis and facial expression in particular. Moreover we address strategies for syntactic markup.

Requirements for a Markup Language in NECA

While typical multimedia markup languages are designed to support a basically text-based annotation of multimodal input to media players, the markup in the NECA project is a means for representing various kinds of expert knowledge required at the different interfaces between the components in the NECA architecture. This requires representation of information at different levels of granularity as well as at different levels of description such as morpho-syntax, information structure, text/speech, facial expression and gesture.

As described in deliverable D1a “Specification of the general system architecture”, the information flow in the NECA architecture proceeds in a pipeline. From a scene generator, which amongst other things gets input from an affective reasoning component, to a multimodal generator, to a speech synthesis component, to a gesture assignment component, to a media player.

Scene generation: The output of the scene generator is a description of the particular scene to be played at the end of processing specifying the semantic content of a scene, the agents that take part in the scene, the roles they play and the emotions they display. At this stage a collection of verbal and nonverbal acts is available and an event-based timing of verbal and nonverbal acts is specified at a high level of abstraction.

Multimodal natural language generation: The output of the scene generator is input to a component, where text assigned with gesture information at a high level of abstraction is generated, and provisions for prosody assignment in speech synthesis are made, such as specification of parts-of-speech, syntactic structure, and the partition of utterances into theme and rheme.

Speech generation: The speech synthesis uses the output from the multimodal generator to create two types of output: (1) a phonemic transcription of the text which is marked up with prosodic information and information on the duration of the sound segments in milliseconds; (2) sound files of the utterances that will be played in sync with the visual animation of the agents in the scene.

Gesture assignment: Major tasks of the gesture assignment component are fine-tuning the timing between facial expression/gesture and speech, as well as integration with physiologically motivated body expressions such as regular eye blinking or thorax movement due to breathing. The granularity of gesture tags at this stage in processing is strongly influenced by the player technology used. In

order to be flexible with respect to player technology, an extra layer for the mapping of the output of the gesture assignment component to the actual player is required.

A variety of markup languages exists which are either designed for the representation of information at individual levels of description or provide a confederation of markup languages for multi-media annotation. An extensive survey of markup languages for multi-media applications has been published recently by the EALES/ISLE project [Wegener-Knudsen et al. 2002]

In the following, examples of existing markup languages and their feasibility for NECA will be discussed. The selection presented is determined by the NECA-specific requirements.

Multimodal Markup Languages

A number of languages for the markup of the multimodal behaviour of virtual humans/ avatars has been proposed, a selection of which will be described in the following. As synchronization of verbal and nonverbal information is a major issue in NECA, we will have a closer look at timing mechanisms specified for the individual markup languages.

VHML

Outlay and Components

Virtual Human Markup Language (VHML) is an XML/XSL-based markup language for the representation of different aspects of “virtual humans”, i.e. avatars, such as speech production, facial and body animation, emotional representation, dialogue management, and hyper and multi-media information ([Gustavsson et al. 2002] and VHML-homepage <http://www.vhml.org>). VHML is mainly intended for implementations of avatars in WWW-based applications. VHML aims at W3C compliance, but is not a W3C standard. It comprises a number of special purpose languages, namely

- EML (Emotion Markup Language),
- GML (Gesture Markup Language),
- SML (Speech Markup Language), based on SSML which has been defined by the W3C consortium,
- FAML (Facial Animation Markup Language),
- BAML (Body Animation Markup Language),
- XHTML (eXtensible Hyper Text Markup Language),
- DMML (Dialogue Manager Markup Language) based on the W3C dialogue manager

Details on the individual languages can be found in the respective subsections of the section on Markup for Individual Representation Levels.

A special characteristic of VHML is the attempt to combine existing special purpose markup languages into a multimodal markup. This is basically achieved by inheritance of elements: SML,

FAML and BAML inherit from both EML and GML. While SML and FAML also define their own elements, BAML at the current state only inherits elements, i.e. no specific attributes for the animation of the body are currently available.

VHML Timing and synchronization

In VHML timing of animation-elements in relation to each other and in relation to the realisation of text is achieved via the attributes “*duration*” and “*wait*”. These take a time value in seconds or milliseconds and are defined for all elements in EML and FAML, i.e. for those parts of VHML concerned with animation.

“*wait*” specifies the duration of a pause before continuing with other elements or text, i.e. “*wait=1s*” means that the *succeeding* item has to wait 1 second before being played. If “*wait*” is not specified (default) then both elements are realized simultaneously, i.e. the default behaviour is to play animations and text in parallel. Sequential rendering or partial overlap have to use “*wait*”. Though in principle all sorts of temporal ordering can be specified using this mechanism, it has to be noted, that for this purpose the concrete duration of the respective elements has to be known beforehand because both “*wait*” and “*duration*” allow only for fully specified duration values.

Examples:

```
<look-left duration="2s"/><eye-blink duration=40ms/> Hello! ... : Looking left
(for 2 seconds), blinking and saying “Hello!” are started simultaneously.
```

```
<look-left duration="2s wait=2s"/><eye-blink duration=40ms/> Hello! ... :
First looking left (for 2 seconds), then start blinking and saying “Hello!” simultaneously
```

MPML

MPML (Multimodal Presentation Markup Language) is an XML-based markup language developed to enable the description of multimodal presentation on the WWW based on animated characters. MPML is under development by Zong Yuan at Ishizuka Lab (Department of Information and Communication Engineering, University of Tokyo). It is currently available as version 2.0e (Jan 2002) ([Tsutsui et al. 2000], [Zong et al. 2000])

MPML is motivated by the goal to provide an easy to use markup language for writing “attractive” multimodal presentations and it supports functions for controlling verbal presentations and scripting agent behaviors.

MPML offers functionalities for synchronising media presentation (reusing parts of the Synchronized Multimedia Integration Language – SMIL) and basic interactivity (e.g. via a <listen> element). Figure 1 shows the overall layout of an MPML presentation in the WWW. On the client side the MPML script is interpreted either by a special MPML player, an XML-browser (which in turn calls a plug-in) or by a converter that produces a script in a format suitable for an agent system (such as MS-Agents).

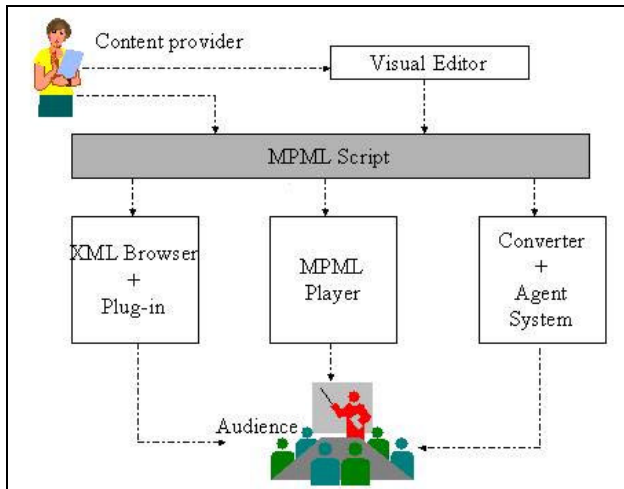


Figure 1 Outlay of a WWW-based MPLM-presentation

Figure 2 presents the whole hierarchy of XML-tags for MPML. For the purpose of this survey we will concentrate on MPML's means for specifying avatar actions. MPML offers the following elements:

- `< speak >` This provides a simple TTS interface: Text within this element will be spoken by a TTS-system. Except selecting speaker, voice and speed no further directives to the TTS-system can be specified.
- `< move >` This specifies the agent to move to a certain point at the screen.
- `< play >` This element is used to handle action via the attribute "act". Possible actions are currently those, which are supported by the agent-player used – MS-Agents in this case. "Useful actions" mentioned by the developers are, e.g., Acknowledge, Alert, Announce, Blink, Confused, Congratulate etc.

`< emotion >` In version 2.0e MPML supports the usage of emotions. MPML uses the 22 emotional types of Ortony, Clore and Collins (the OCC-model [Ortony et al. 1988]). The specified emotions are to be expressed by "performing different actions and changing speech parameters (pitch, volume, speed, emphasis of certain words)" [Zong et al. 2000] though the actual mapping from OCC-categories to these parameters is not specified yet.

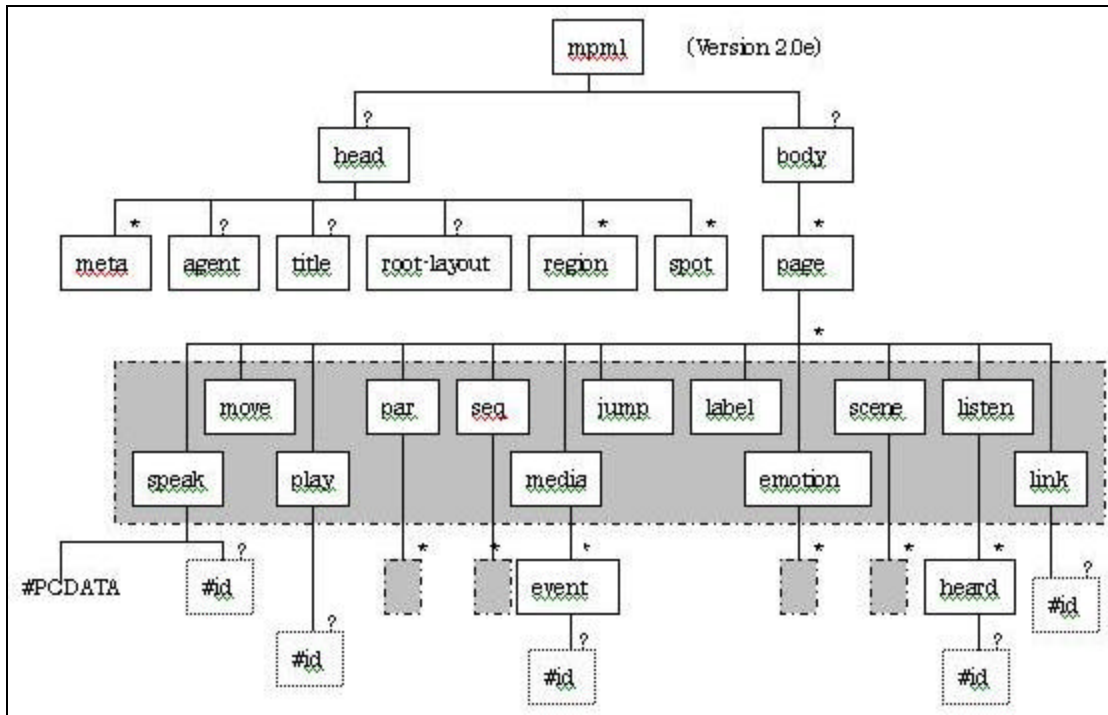


Figure 2: Overview on the structure of XML tags in MPML. “*” denotes Kleene’s star.

Online References:

MPML Homepage: <http://www.miv.t.u-tokyo.ac.jp/MPML/en>

TVML

TVML – TV program Making Language – is a text-based scripting-language, which has been designed to describe a complete television program at a high level of abstraction. Thus TVML is also feasible for a wide range of multimedia and interactive applications. A TVML script is to be translated by a TVML player into real time video and audio.

There are 11 event types in TVML: character, camera, set, prop, light, movie, title, sound, super, narration and video – i.e. facilities to control characters, camera movements, lighting, the inclusion of sound and video effects, the display of titles etc.

Here we will concentrate on character-related control commands. Table 1 shows the whole set of “CG character commands”.

casting(name)	Name CG character
openmodel(modelname,filename)	Open the CG character modeling data
closemodel(modelname)	Close the CG character modeling data
bindmodel(name,modelname)	Bind the CG character modeling data to the character
setvoice(name,voicetype)	Set up the CG character's talking voice

visible(name,switch)	Show and Hide the CG character
position(name,x,y,z,d,posture)	Position the CG character (standpoint)
talk(name,text,emotion,pausehead,pausetail,rate,pitch,intonation,volume,wait)	CG character's dialogue
talkfile(name,filename,emotion,pausehead,pausetail,wait)	CG character speaks prerecorded dialogue
walk(name,x,y,z,d,stopmode,pitch,compass,stop,wait)	CG character walks
stop(name, wait)	Walking CG character stops
sit(name, speed, hiplevel, wait)	CG character sits
stand(name, speed, wait)	CG character stands up
turn(name, speed, style, wait)	CG character faces different direction
bow(name, style, speed, level, wait)	CG character bows
look(name, what, track, speed, wait)	CG character looks at something
gaze(name,pitch, yaw, roll, speed, wait)	CG character's head turns in specified direction
shake(name,state, level)	CG character shakes
openmouth(name,state, level)	CG character's mouth opens
openkeyframe(keyframename, filename)	Open keyframe data file
closekeyframe(keyframename)	Close keyframe data file
keyframe(name, keyframename, speedratio, repeat, wait)	Operate CG character by keyframe data

Table 1 TVML-labels for controlling characters

As can be seen from the set of character labels, TVML offers the following functionalities for its characters:

- Selecting and deselecting characters: casting(), openmodel(), closemodel(), bindmodel()
- Simple text-to-speech interface: setvoice(), talk().
- Moving around and postures: walk(), stop(), sit(), stand(), etc.
- Simple facial animation: openmouth(), closemouth(), look(), gaze()
- Replay of keyframe animations and prepared speech-files: openkeyframe(), talkfile() etc.

Online References:

A TVML editor for Windows 95/98 (Japanese version) is available at:

<http://www.strl.nhk.or.jp/TVML/English/Esitemap.html>

The TVML-homepage: <http://www.str1.nhk.or.jp/TVML/English/E03.html>

Markup for Individual Representation Levels

Syntactic Markup

Syntactic information, in particular part-of-speech, syntactic structure and grammatical functions, is important for prosody generation in speech synthesis. In the NECA demonstrators speech synthesis is performed by a concept-to-speech module. The input to the speech-synthesis is not plain text, but text enhanced with labels carrying syntactic and grammatical information. This is due to the fact that the texts to be uttered have been produced by the natural language generation module.

Information on syntax and part-of-speech status is to be used mainly for the purpose of facilitating the generation of appropriate prosody within the speech-synthesis module. We, therefore, conducted a survey into how information on syntax is included in markup-languages for speech synthesis (See next section). The survey showed that among the speech markup-languages only Microsoft's Speech Application Programming Interface (MS-SAPI) provides (optional) tags for syntactic annotation.

Part of Speech Tags in Microsoft SAPI

Only information on the part-of-speech status can be specified in SAPI and the respective attribute `<PartOfSp>` only allows for the following 6 values: "Unknown", "Noun", "Verb", "Modifier", "Function", and "Interjection".

Whether such a very coarse-grained classification is sufficient for the purpose of prosody generation in speech synthesis needs to be evaluated. The restriction to encode part-of-speech information only and omit both syntactic structure and grammatical function seems to be too limited and is therefore not recommended for NECA.

Syntactic Markup for Large Text Corpora

In this section we present two well-known and widely used annotation schemes for the markup of large text corpora: the Penn Treebank annotation scheme and the NEGRA Corpus annotation scheme (see <http://www.cis.upenn.edu/~treebank/home.html> and <http://www.coli.uni-sb.de/sfb378/negra-corpus>).

The former has been designed for large-scale annotations of English text, the latter for German. Schemes for syntactic annotation typically comprise three levels of representation: part-of-speech, syntactic category and grammatical function. The granularity of the information represented in the tag sets corresponding to the representation levels strongly depends on the application area. For large scale annotation of training corpora for stochastic NLP, like the Penn Treebank and the NEGRA Corpus, tag sets need to be moderate in size in order to avoid the sparse data problem of statistics. As the corpus annotation task can be costly, in terms of both time and money, the information represented in the annotation is kept as general and theory independent as possible, which makes

these kinds of annotation an appropriate starting point for the development of the syntactic markup part of the NECA RRL.

In the following we give examples of tags used in the NEGRA Corpus and the Penn Treebank.

Part-of-speech tags: represent morpho-syntactic information at word level. The NEGRA Corpus comprises 55 part-of-speech tags, the Penn Treebank has 48 tags. Part-of-speech tagsets typically contain tags such as article, common noun, proper noun, preposition, adjective, adverb; some verb tags; tags for different kinds of pronouns, particles, punctuation, tags for foreign material and nonwords. As an example the noun and verb tags used in the NEGRA Corpus and the Penn Treebank are listed below.

Part-of-speech tags for nouns and verbs, NEGRA Corpus

NN	common noun
NE	proper noun
NNE	combination of proper and common noun
VVFIN	finite verb, main
VVIMP	imperative, main verb
VVINFINF	infinitive, main verb
VVIZU	infinitive mit „zu“, main verb
VVPP	past participle, main verb
VAFIN	finite verb, auxiliary
VAIMP	imperative, auxiliary
VAINFINF	infinitive, auxiliary
VAPP	past participle, auxiliary
VMFIN	finite verb, modal
VMINFINF	infinitive, modal
VMPP	past participle, modal

Part-of-speech tags for nouns and verbs, Penn Treebank

NN	noun, singular or mass
NNS	noun, plural
NP	proper noun, singular
NPS	proper noun, plural
VB	verb, base form
VBD	verb, past tense
VBG	verb, gerund or present participle
VBN	verb, past participle
VBP	verb, non-3rd person singular present
VBZ	verb, 3rd person singular present

Syntactic category tags: represent information at phrase level, such as S, NP, VP, etc. The major differences between the NEGRA and the Penn tagset are due to the different approaches to grammatical structure. While in the Penn Treebank bar-level structures are represented and discontinuous constituents are annotated by means of a trace-filler mechanism, in the NEGRA Corpus no bar-levels are represented and discontinuous constituency is represented via crossing edges in the tree structure. In the following complete lists of the category tags available in NEGRA and the Penn Treebank scheme are presented.

Syntactic category tags – NEGRA Corpus

1	NP	noun phrase
2	AP	adjective phrase
3	PP	adpositional phrase
4	S	sentence
5	VP	verb phrase (non-finite)
6	VZ	„zu“-marked infinitive
7	CO	coordination
8	AVP	adverbial phrase
9	AA	superlative phrase with "am"

10	CNP	coordinated noun phrase
11	CAP	coordinated adjective phrase
12	CPP	coordinated adpositional phrase
13	CS	coordinated sentence
14	CVP	coordinated verb phrase (non-finite)
15	CVZ	coordinated zu-marked infinitive
16	CAVP	coordinated adverbial phrase
17	MPN	multi-word proper noun
18	NM	multi-token number
19	CAC	coordinated adposition
20	CH	chunk
21	MTA	multi-token adjective
22	CCP	coordinated complementiser
23	DL	discourse level constituent
24	ISU	idiosyncratic unit
25	QL	quasi-language

Syntactic category tags – Penn Treebank

1	ADJP	adjective phrase
2	ADVP	adverbial phrase
3	AUX	auxiliary
4	CONJP	multi-word coordinating conjunction
5	CP	phrase coordination
6	FRAG	sentence/clause fragment
7	NP	noun phrase
8	PP	prepositional phrase
9	RRC	reduced relative clause

10	S	sentence
11	SBAR	relative clauses, subordinate clauses, including indirect questions
12	SBARQ	wh-question
13	SINV	inverted sentence, e.g. <u>Willie caught the ball, said Casey.</u>
14	SQ	question, subject and auxiliary are missing
15	UCP	coordination of unlike phrases
16	VP	verb phrase
17	WHADJP	only used if there is wh-movement, they always leave a trace
18	WHADVP	why, only used if there is wh-movement, they always leave a trace
18	WHNP	only used if there is wh-movement, they always leave a trace, e.g. who, what, which
19	WHPP	only used if there is wh-movement, they always leave a trace
20	QP	quantifier phrase, e.g. <u>more than 7</u> cars

Grammatical function tags: represent information on the grammatical function of constituents in a larger phrase. Typical grammatical functions are head, modifier, complementizer, subject, accusative object, dative object, predicate, etc. The NEGRA annotation scheme provides a vast set of function tags, i.e., 46 tags in total, compared to the Penn Treebank scheme where only some 16 function tags are available. To some extent this difference can be explained by the differences in word order variation between German and English. Especially for English arguments the position within a sentence is fixed.

Which particular information on part-of-speech, syntactic category and grammatical functions will be required for prosody generation in NECA, and how far we will be able to use the same set of function tags for English and for German applications of the NECA platform still needs to be explored.

Speech Synthesis Markup

Speech Synthesis Markup Language (SSML)

The Speech Synthesis Markup Language (SSML) is a standard currently under development by W3C's Voice Browser Working Group. The actual version is "W3C Working Draft 5 April 2002". SSML is designed to provide an XML-based markup language for the generation of synthetic speech both in the WWW as well as in stand-alone synthesizers. It aims to give authors of synthesizable text the opportunity to control the output of synthesized speech with respect to pronunciation, volume, pitch, rate, etc. across different platforms capable of synthesis.

The development of SSML is strongly based on other existing speech markup languages –especially JSML (Java Speech Markup Language) and SABLE¹. These will thus not be surveyed here.

The current draft of SSML comprises 12 elements which are grouped as follows:

Document Structure, Text Processing and Pronunciation

<speaK>: The root element for SSML documents.

Attribute:

- lang : Specifies the language to be used (lang is also defined for <paragraph> and <sentence>)

<paragraph> and <sentence>: Represents the internal structure of texts

<say-as>: Gives information on the “type of text” included in this element, in order to aid the correct PRONUNCIation by specifying that a text is to be interpreted, e.g., as currency, date, address.

Attribute:

- type : Encompasses many different types, like pronunciation type (“acronym”, “spell-out”) numerical type (“number”, “ordinal”, “cardinal”, “digits”), time and measure types (“date”, “time”, “currency”, “measure”, ...)

<phoneme>: Provides phonetic pronunciation.

Attribute:

- alphabet: Specifies which phonetic alphabet is used (e.g., “ipa”)

<sub>: When synthesizing substitutes the contained text by the one given in the “alias” attribute

Attribute:

- alias: specifies the text to be pronounced instead (e.g. _{WWW})

Prosody and Style

<voice>: Specifies the voice to be used

Attributes:

- lang (optional language specification)
- gender (“male”, “female”, “neutral”)
- age (preferred age of the voice to speak the contained text – integer)

¹ Note, that in the literature on SABLE it is claimed, that SABLE is based on SSML, while in the W3C-SSML documents it is noted that SSML is based on SABLE. This confusion is due to the fact, that there exists an older – now obsolete – markup-language called “SSML”, which then was replaced by SABLE. (Taylor P., Isard A.: SSML: A speech synthesis markup language, Speech Communication, (21), pp.123-133, 1997.)

- name (platform-specific voice name)
- variant (indicating a preferred variant of the selected voice – integer)

<emphasis>: The contained text is to be spoken with emphasis.

Attribute:

- level: strength of emphasis (“none”, “reduced”, “moderate”, “strong”)

<break>: An empty element controlling pausing and realisation of prosodic boundaries

Attributes:

- size (strength of boundary (“none”, “small”, “medium”, “large” – optional)
- time (duration of a pause in seconds or milliseconds – optional)

<prosody>: Permit control over the prosody to be used

Attributes:

- pitch (the baseline in Hertz, a relative change or the values “default”, “low”, “medium”, “high”)
- contour (specify a concrete pitch contour)
- range (the pitch range in Hertz, a relative change or the values “default”, “low”, “medium”, “high”)
- rate (the speech rate in words per minute. A relative change or the values “default”, “slow”, “medium”, “fast”)
- duration (the time to be used to pronounce an item in seconds or milliseconds)
- volume (in a range from 0.0 to 100.0, a relative change or the values “default”, “silent”, “soft”, “medium”, “loud”)

Other Elements

<audio>: Embed an audio file, which is replayed when the element is reached

Attribute:

- src (specify the name of the audio file)

<mark> Place a marker in the text to be used either for internal reference within the SSML document or to be used externally by other documents. When speech synthesis reaches a <mark> element it issues an event with its name

Attribute:

- name (string issued as event name when mark is reached – required)

Online References:

The actual version of SSML is always to be found at: <http://www.w3.org/TR/speech-synthesis>

Microsoft SAPI TTS XML

Within its Speech Application Programming Interface (SAPI), Microsoft offers its own interface to speech synthesis. The current version is SAPI 5.1. Though SAPI itself is not a markup language SAPI 5.1 offers an XML based language for TTS which is explicitly inspired by SABLE (one of SSML's ancestors) but is not aiming for real compatibility with SABLE. As it has to be taken into account, that Microsoft's power on the market tends to apply some pressure on every standardization effort, we will perform a short comparison of SAPI 5.1. TTS XML with SSML.

Roughly speaking, many of the differences are only minor deviations in terminology (e.g. `<ssml:say_as>` and `<sapi:context>` are basically identical in function). Some differences arise from the fact that a functionality is expressed in terms of its own element in the one markup-language but in terms of an attribute in the other (e.g. `<ssml:say_as type="spell">` and `<sapi:spell>` have the same effect). Generally speaking the major differences in functionality can be summarized as follows: SABLE offers finer-grained control for the specification of factors influencing prosody (e.g., several levels of strength in `<emph>`, several levels of size in `<break>`) as well as concrete control over the acoustic parameters (i.e., fundamental frequency in Hertz and duration in milliseconds) for specifying prosody.

Online References:

<http://msdn.microsoft.com/library/default.asp?url=/library/en-us/sapi/Welcome.asp>

SML in VHML

VHML (Virtual Human Markup Language) is an attempt to combine existing markup-languages developed for the various aspects of human-computer interaction (e.g. facial expression, body animation, emotional representation) into a unified specification language. The sub-part of VHML concerned with the markup for speech synthesis is called Speech Markup Language (SML) and is – according to the current “VHML Working Draft v0.3” from 21.Oct.2001 – based on W3C's SSML. Comparing SML to the current version of SSML points out, that roughly speaking SML currently is a slightly downsized variant of SSML. The most important difference to SSML and SAPI is, that it foresees the labelling of the speaker's emotion via VHML's Emotional Markup Language (EML). Emotion-tags specified in EML are inherited by SML and are thus visible to the speech synthesis.

Comparison of SSML, SML and SAPI

In Table 2 a rough comparison of functionality and tag-sets of SSML, its VHML-derivate SML, and MS-SAPI is performed. “N.A.” denotes “Not available”. Elements and their meaning are usually described in more detail in the section on SSML.

Functionality	SSML	SML	SAPI	Remarks
---------------	------	-----	------	---------

Functionality	SSML	SML	SAPI	Remarks
Root element	<speak>	N.A.	<sapi>	
Defining language	“lang” attribute for <speak> <sentence> and <paragraph>	N.A.	<lang>-element or “lang”-attribute of <voice> element	
Structuring of text	<paragraph> <sentence>	<paragraph>	N.A.	<vhml:paragraph> is not part of SML proper, but available in VHML
Specifying how a certain content is to be interpreted. Attributes are, e.g., <i>email</i> , <i>number</i> , <i>ordinal</i>	<say-as>	<say-as>	<context>	
Specify, that contained text is to be spelled out (e.g. “USA”)	<say-as type=spell- out>		<spell>	
Provide part of speech information	N.A.	NA	<PartOfSp>. Values are: “Unknown”, “Noun”, “Verb”, “Modifier”, “Function”, “Interjection”	
Provide phonetic pronunciation	<phoneme>	<phoneme>	<pron>	
Indicate that a specified text substitutes another	<sub>	N.A.	N.A.	
Specify/change the voice used	<voice> Attributes are lang, gender, age, variant, name	<voice>	<voice>	

Functionality	SSML	SML	SAPI	Remarks
Mark content as emphasized	<p><emphasis></p> <p>Attribute: level [Values: “strong”, “moderate”, “none”, “reduced”]</p>	<p><emphasize-syllable></p> <p>extends SSML with Attribute: affect [Values “pitch”, “duration”, “both”] and target [the phoneme to be emphasized]</p>	<p><emph></p>	<p>In SAPI only <EMPH> available w/o further differentiation.</p> <p>In SML the syllable and optionally the prosodic means to signal emphasis and the exact position of the phoneme to be emphasized can be specified.</p>
Control of pausing/prosodic boundaries.	<p><break></p> <p>Attribute: size [Values: “none”, “small”, “medium”, “large”]</p> <p>Attribute: time (optional): duration of pause</p>	<p><break></p>	<p><silence msec=100></p> <p>Inserts silence of 100ms length</p>	<p>In SAPI only length of pause specified – no further differentiation</p>
Control prosody.	<p><prosody></p> <p>Attributes: pitch, contour, range, rate, duration, volume</p>	<p><prosody></p>	<p><pitch></p> <p><rate><speed></p> <p><volume></p>	<p>SSML allows for a finer grained control of prosody. Equivalents to “contour” and “duration” are missing entirely in SAPI</p>
Insert arbitrary audio file (e.g., recorded speech, music)	<p><audio></p>	<p><embed></p>	<p>NOT AVAILABLE</p>	<p><vhtml:embed> is not part of SML proper but of VHML – it allows for embedding of foreign filetypes in general</p>
Specify emotion of speaker	<p>N.A.</p>		<p>N.A.</p>	
Place a marker in the text: An event will be issued by the synthesizer when reaching the mark	<p><mark></p>	<p><mark></p>	<p><bookmark></p>	<p><ssml:mark> can have content, <sapi:bookmark> has to be empty</p>

Table 2 Summary of all attributes in SSML, SML, and SAPI

This table once again reveals the close similarities between SSML and SML. SAPI basically offers a sub-set of SSML's labels. Many of the remaining differences are only syntactic.

Facial Animation Specifications

FAML Facial Animation Markup Language

FAML comprises 20 elements defining movements of the eyes, eyebrows and head, as well as eye blinking and jaw opening and closing. Being part of VHML, all EML and GML elements are inherited. The following attributes can be assigned to the FAML elements:

- duration
- intensity
- mark (an attribute common to all sub-parts of VHML: can be used to set an arbitrary mark; an engine can report an event when this mark is reached)
- wait (is used for synchronisation)

The set of FAML elements:

look	eyes	Head	head-roll	eyebrow	eye-blink	wink	jaw
-right	-right	-right	-right				open-
-left	-left	-left	-left				close-
-up	-up	-up		-up			
-down	-down	-down		-down			

Table 3 Elements in FAML

This table has to be interpreted as FAML containing the following attributes for specifying gaze-direction: <look-right> <look-left> <look-up> and <look-down>

The effect of the respective FAML-elements:

- <look-...>: Turn both eyes and head to the specified direction, eyes and heads move with the same rate.
- <eyes-...> Only the eyes turn to the specified direction. Head stays in position.
- <head-...> Only the head turns to the specified direction. Eyes stay in position
- <head-roll-...> Roll head.
- <eyebrow-...> lift or lower eyebrow (optional attribute “which” with values left, right and both)
- <eye-blink> Blink (with both eyes)
- <wink> Wink with one eye (optional attribute “which” with values left and right)
- <open-jaw> <close-jaw>

Facial Action Coding Scheme (FACS)

Facial Action Coding Scheme (FACS) [Ekman & Friesen 1978] is a scheme developed to manually measure facial expressions – and is intended to be able to encode every visible change in the human face, e.g. movements of skin, wrinkles, folds, deformation of shapes. This is accomplished by decomposing these changes into minimal visual actions – so called Action Units (AUs). There are 66 AUs defined in FACS which are related to actions of single muscles (e.g., AU1: Inner Brow Raiser) or groups of muscles (e.g., AU5 “Head Tilt Left”) and they describe direct effects as well as secondary effects (propagation of wrinkles and folds) of muscle movements.

Several AUs can be – or in many cases have to be – combined in order to describe a change in the face. E.g. “surprise eyebrow” is described as a combination of AU1 (Inner Brow Raiser) + AU2 (Outer Brow Raiser), which should result in highly raised, curved eyebrows. The single AUs are also assigned with an intensity value (low, medium or high).

FACS is a scheme for manually encoding facial expressions, originally developed to facilitate psychological studies of human expression. Nevertheless the concept of Action Units, i.e. the decomposition of changes in the human face into minimally perceptible units, which are (not strictly) anchored to muscular actions, has strongly influenced the development of coding schemes used for facial animation.

Minimal Perceptible Action MPA

Minimal Perceptible Actions (MPAs) – developed in the early 90s [Kalra 1991] – presents another approach for the representation of facial expression. Inspired by Ekman and Friesen’s Action Units [Ekman & Friesen 1978]. MPAs are defined on the basis of muscles or points representing muscles. 65 MPAs have been defined covering low, mid and high level facial expressions such as *open mouth, head turn, raise corner lip*. This coding approach had a strong influence on the facial animation part of MPEG-4 by which eventually superseded it.

MPEG -4 Facial Animation

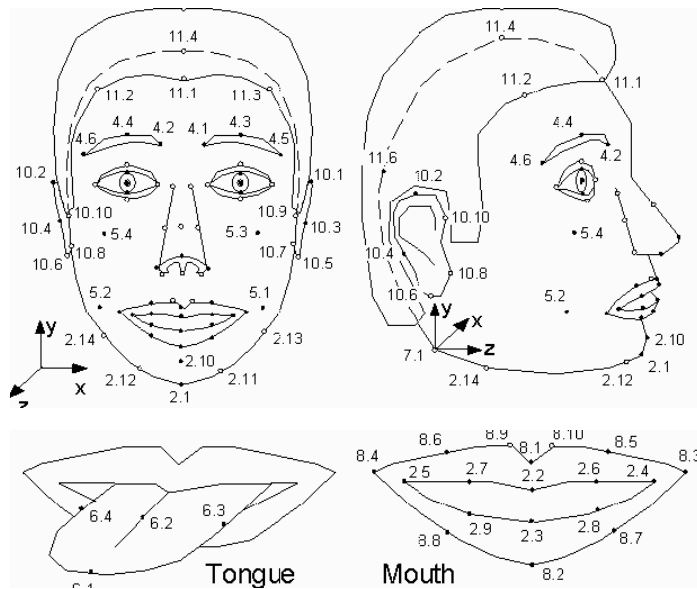
MPEG-4 is an ISO/IEC multimedia compression standard developed by MPEG (Moving Picture Experts Group) for encoding different audiovisual objects. As for the representation of synthetic visual objects – which include all sorts of animation objects – MPEG-4 is based on the prior VRML (Virtual Reality Markup Language) i.e. it uses the same mechanisms for representation and manipulation of 3D objects.

In order to support talking head applications MPEG-4 offers the means to allow facial animation at a very low bit rate. Animation of talking heads in MPEG-4 is based on the following components: a model of the face in its neutral state, a number of feature points (FP) on this neutral face as reference points, and a set of facial animation parameters (FAPs) [Tekalp&Ostermann 2000].

The neutral face is defined as a face with all muscles relaxed, mouth closed, gaze in the direction of the Z-axis. On this neutral face MPEG-4 specifies 84 feature points (FPs) the position of which has to be known for any MPEG-4 compliant face model. Feature points are arranged in groups like cheeks, eyes, and mouth. Some of these FPs (e.g. those defining the form of the hairline) are not

affected by animation. The purpose of the majority of FPs though is to provide spatial references for defining facial action parameters (FAPs).

MPEG-4 specifies a set of 68 FAPs. There are two *high-level* animation parameters, namely FAP 1 “viseme” (used for the visual representation of phonemes) and FAP 2 “emotion” (allowing for the specification of affective states “Anger”, “Joy”, “Disgust”, “Sadness”, “Fear” and “Surprise”). These define possibly complex displacements of different groups of FPs². The remaining *low-level* FAPs define the displacement of one FAPs or group of FAP along a single dimension (X, Y, Z axis). The *value* of a FAP defines the magnitude of the movement in relation to the neutral face.



In order to allow for the animation of faces of different size and form, the values of FAPs are specified in terms of animation parameter units (FAPU). These are also computed from distances of major FPs in the neutral face and allow for a scalability of facial animations. Table 4 contains the definition of FAPUs. Table 5 shows examples of FAPs and the usage of FAPUs.

Animation of a face in MPEG-4 is achieved by specifying the respective FAP values at each time instant, i.e. for each animation frame. In order to be able to interpret such a stream of FAP-values a MPEG-4 player has to have model specific animation rules to produce the facial action corresponding to each FAP. Every MPEG-4 player already contains a face model, i.e., a MPEG-4 animation can be interpreted by the player without the need for specifying a proprietary face. In case the encoder wants to animate another face than the player’s standard face, a face model has to be downloaded to the decoder first. This is performed by specifying Facial Definition Parameters (FDPs), a complex set of parameters which defines the shape of the neutral face and specifies for every FAP how they affect the movement of FPs and how the movement of a FP affects its neighbouring vertices.

² E.g., emotion “Joy” is defined as “The eyebrows are relaxed. The mouth is open and the mouth corners pulled back toward the ears.”

IRISD0	Iris diameter	$IRISD = IRISD0 / 1024$	
ES0	Eye separation	$ES = ES0 / 1024$	
ENS0	Eye : nose separation	$ENS = ENS0 / 1024$	
MNS0	Mouth : nose separation	$MNS = MNS0 / 1024$	
MW0	Mouth width	$MW = MW0 / 1024$	
AU	Angle unit	10E-5 rad	

Table 4 Facial Animation Paramters (FAPUs) and their definitions

#	FAP name	FAP description	Units	Uni- or Bidir	Pos Motion
1	Viseme	Set of values determining the mixture of two visemes for this frame (e.g. pbm, fv, th)	na	na	na
2	Expression	A set of values determining the mixture of two facial expression	na	na	na
3	Open_jaw	Vertical jaw displacement (does not affect mouth opening)	MNS	U	down
4	Lower_t_midlip	Vertical top middle inner lip displacement	MNS	B	down
31	raise_l_i_eyebrow	Vertical displacement of left inner eyebrow	ENS	B	up
37	squeeze_l_eyebrow	Horizontal displacement of left eyebrow	ES	B	right

Table 5 Examples MPEG-4 FAPs and their definition.

MPEG-4 is a standardized multimedia compression format that offers facial animation at a very low bit rate. It may be an interesting option for the player technology in the future. Comparing it to animation specifications like FAML (VHML) it becomes clear, that MPEG-4 operates at a different level of abstraction. E.g. a “simple” action like <eye-blink> in FAML has to be specified as a bitstream in MPEG-4 where for every animation frame the change of values of a number of FAPs involved in the movement of eye-lids has to be specified.

Online References:

The complete table of MPEG-4 FAPs can be found at:

<http://www-dsp.com.dist.unige.it/~pok/RESEARCH/MPEG/fapspec.htm>

Introduction to MPEG-4: <http://mpeg.telecomitalialab.com/standards/mpeg-4/mpeg-4.htm>

ToonFace / CharToon

ToonFace [Thorisson:1996] is an animation framework that allows for the coding of non-photo-realistic facial expressions in limited detail, but its simplicity allows for the creation of 2D animations in rather short time. CharToon is an extended Java-based version developed within the European project FASE between 1997 and 2000.

We will concentrate on ToonFace here in order to capture the underlying principles of both systems.

The ToonFace system consists of an Editor (Apple Macintosh based) and an Animator i.e. a player (running on an SGI-workstation only).

In order to keep the model simple and thus allow for easy and fast processing ToonFace restricts its models to the usage of only five kinds of polygon and four kinds of polygon manipulation and it employs very simple linear interpolation for animation.

In ToonFace the face is divided into seven main features, namely 2 eye brows, 2 eyes, 2 pupils and the mouth. The eye brows have 3 control points each., eyes and mouth have 4 and the pupils 1. The main idea of ToonFace is to restrict the number of control points in a face as well as their degrees of freedom to a minimum. Thus e.g. only 2 of the 4 control points for the eyes are manipulated, the other two stay fixed. The complete set of control points that can be moved are the following 17 points:

- **Brows:** Brl = brow/right/lateral; Brc = brow/right/central; Brm = brow/right/medial]; With similar labels for the *left* brow: Bll; Blc; Blm;
- **Eyes:** Eru = eye/right/upper; Erl = eye/right/lower; Accordingly: Elu; Ell
- **Pupils:** Prh = pupil/right/horizontal Prv = pupil/right/vert; Accordingly: Plh;Plv
- **Mouth:** Mrh = mouth/right/horizontal; Mrv = mouth/right/vertical; Accordingly: Mlh Mlv; Mb = mouth/bottom
- **Head:** Hh = head/horizontal; Hv = head/vertical

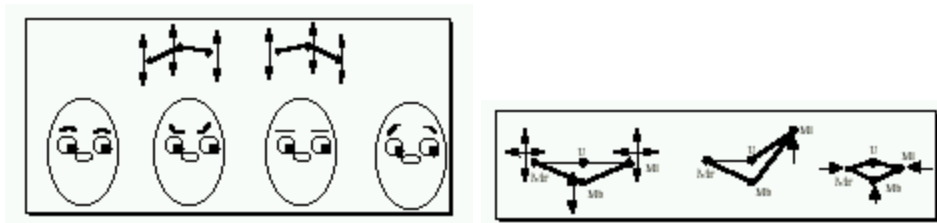


Figure 3: Effects of changing the 3 control points (CP) of brows (left) and of the 3 CPs of the mouth (right). Arrows indicating number and direction of the degrees of freedom)

ToonFace allows to use arbitrary free polygons for designing faces, as long these parts are not manipulated (e.g., hair, hats). The other possibility is to use polygons attached to a whole feature (e.g., eye, brow) or a single control point. In this case movement of the respective control points will result in a distortion of the polygon.

Once a face model is created in the Editor and loaded into the Animator, the face can be animated by sending commands of the following format:

```
CODE Controlpoint direction abs-pos exec-time
```

This specifies to move a particular control point in either a horizontal or vertical direction to an absolute position `abs-pos` which takes a certain amount of time in milliseconds (`exec-time`). ToonFace also foresees an interface to a speech synthesis module (DEC-Talk) in order to produce talking heads.

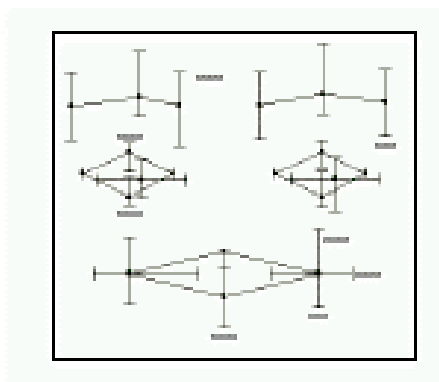


Figure 4 Control points for eyebrows, eyes and mouth, their degrees of freedom and ranges of movement

CharToon, developed at CWI in Amsterdam between 1997 and 2000 [Noot&Ruttkay 2000], incorporated many ideas of the somehow outdated ToonFace. Some of the enhancements: In addition to a face editor and a player it also includes an Animation Editor which allows editing of timing tracks for the animation of different control points in a GUI. In version 2.1 [Ruttkay&Lelievre 2000] it also offers a library of pre-defined macros for specifying emotional states and visemes. Because it is written in Java (1.1.), it is less hardware dependent than the Apple/SGI based implementation of ToonFace.

Online References:

CharToon Homepage: <http://www.cwi.nl/projects/FASE/CharToon>

Emotional Markup

Many of the “higher-level” markup languages presented in this survey provide some means to specify the emotional state of an avatar. In general these labels are intended to influence its facial expression, the emotional shading of the avatar’s speech, and in some cases also its gesture and posture.

In Table 6 a comparison of the emotion tags offered by the different markup-languages is performed.

It reveals, that there is quite a consensus about the number and type of labels between EML (VHML's emotion-component), MPEG-4 and CharToon. TVML in contrast is very restricted, only distinguishing between "normal" and "excited". MPML uses the most elaborate markup-scheme, though the interpretation of these labels is *not* specified yet, i.e., it is left open how a specific emotional tag is to be rendered in terms of visual and auditory appearance of the avatar.

CharToon's emotional tags, provided in the table, have to be interpreted in a different way though. A single emotional tag, in fact, is just a name for a whole group of specific animation-parameter macros. The repertoire of *Anger*, e.g., includes the following animation files:

Be_Careful_What_You_Say.all, Teased_Pested.all, Furious.all, Annoyed.all, Reproach.all, Rage_Hate.all, Deceived_BadMood.all and *Sulk.all*.

EML (VHML)	MPEG-4	CharToon	TVML	MPML (OCC-model)	
Speech, Face, Gestures	Face	Face	Speech, Face	Speech, Face, Gestures	
Neutral	(implicit?)	(Neutral)	Normal	(implicit?)	
Afraid	Fear	Fear		Fear	
Angry	Anger	Anger			
Disgusted	Disgust	Disgust			
Confused					
Dazed					
Happy	~ Joy			~ Happy-for	
Sad	Sadness	Sadness		~ Distress	
Surprised	Surprise	Surprise			
Remaining tags. Not directly related to any of the other markup-schemes:					
		Smile Other: Crying_of_joy, Bored, Impressed, Not_Sure, ...		Exited	Admiration, Disappointment, Disliking, Fears-confirmed, Gloating, Gratification, Gratitude, Hope, Joy, Liking, Pride, Relief, Remorse, Reproach, Resentment, Satisfaction, Shame, Sorry-for

Table 6 Overview and comparison of emotion-labels in different markup languages. "~" denotes not exact or questionable match.

Miscellaneous Markup Languages for Web-based Applications

In addition to the markup-languages specifically devoted to at least some aspects of specifying virtual humans there also exist more general representation languages and standards for Web-based multimodal applications. In this section two of these standards will be presented here, which may prove to be interesting for the purposes of NECA.

VRML Markup for 3D Objects and Spaces

VRML (Virtual Reality Modelling Language) is an ISO-standard for the visualization of 3D objects in virtual environments for the World Wide Web. Particular features of VRML are its feasibility for hyper-linking and 3D navigation. VRML is not a general purpose programming language but closely tied to 3D/multimedia applications.

The VRML consortium has been renamed in the Web3D consortium (<http://www.web3d.org>). Under Web3D X3D has been developed. X3D is an XML-based successor of VRML, which has been launched as an open-standard in August 2001. A subset of X3D is now going to be incorporated into MPEG-4. X3D is VRML97 compatible. VRML97 is an ISO standard. ISO standardisation of X3D is expected in the near future.

A number of VRML/X3D browsers exists. As VRML97 is an ISO standard a variety of authoring tools allow output to VRML or conversion to VRML. However, the vast majority of tools only allow for static modelling, thus VRML-based modelling of interaction is quite a problem. For an overview of VRML compliant browsers and modellers see <http://www.csv.ica.uni-stuttgart.de/vrml/linuxtag/>.

Within H-ANIM (Humanoid Animation Working Group) VRML 2.0³-specifications for a humanoid have been defined. In H-ANIM the human body is defined by a number of segments such as forearm, hand, foot, hip, knee, ankle etc. In addition, a number of viewpoints can be defined which provide the user with the humanoid's eye-view of its world. Features such as the appearance of multiple humanoids in the same file have been addressed as important, but do not yet belong to the scope of the H-ANIM specification. For further information see <http://www.h-anim.org>.

The actual status seems to be that strong efforts are being made to include the specifications of H-anim into MPEG-4/7 in order to supersede VHML in the future.

³ VRML 2.0 currently equals VRML97

SMIL

The Synchronized Multimedia Integration Language (SMIL, pronounced "smile") is a W3C recommendation for a XML-based language for writing interactive multimedia presentations. Its current version is SMIL 2.0, which was released on 07 August 2001.

With SMIL 2.0 an author can describe the layout of presentations on a screen, associate media objects with hyperlinks and – probably most interesting with respect to the purposes of NECA – define the temporal behaviour of presentations. Simply put, it enables authors to specify *what* should be presented *where* and *when*. It enables them to control the precise time that a sentence is spoken and make it coincide with the display of a given image appearing on the screen and it includes provisions for simple interactivity such as control buttons for stop, fast-forward and rewind.

SMIL 2.0 is explicitly intended for being reused within other XML-based languages, in particular for the purpose of dealing with timing and synchronization. Therefore SMIL is organized into a number of markup modules, which define the semantics and an XML syntax for certain areas of SMIL's functionality. Modules are then to be integrated into other languages via profiling i.e. by combining these modules, in order to provide the functionality required by a particular application. SMIL 2.0 comprises a vast number of modules, which can be grouped into bigger domains (e.g., Layout Modules, which are used to define and arrange regions for the display of animations on the screen).

In this summary we centre on the Media Object Modules and the Timing and Synchronization Modules.

Media Object Modules

SMIL Media Object modules are composed of a BasicMedia module and five modules with additional functionality (e.g. MediaClipping, MediaDescription) that build on top of it.

The BasicMedia module defines the baseline functionality of a SMIL player. This module defines the following elements, which allow the inclusion of media objects⁴ into a SMIL presentation:

- ref (generic media reference)
- animation (vector graphics or other animated format)
- audio (audio clip)
- img (still image, such as PNG or JPEG)
- text and textstream
- video (video clip)

Examples for an audio and an image object:

```
<audio id="song1" src="song1.au"/>
```

⁴“Media” in the sense of SMIL comprise both continuous media (e.g. audio and video files or other media for which there is a measurable and well-understood duration) and discrete media (e.g. image files, text files).

```

```

SMIL Timing

SMIL timing defines elements and attributes to coordinate and synchronize the presentation of media objects. SMIL provides three synchronization elements, referred to as *time containers*, to support common timing use-cases: They group their contained children together.

- `<seq>` plays the child elements one after another in a sequence.
- `<par>` plays child elements as a group in parallel.
- `<excl>` plays one child at a time, but does not impose any order.

Media objects and time containers are grouped using these elements. In addition to these elements SMIL timing provides a number of attributes for specifying timing behaviour. Elements have a *begin*, and a simple duration. The *begin* can be specified in various ways, e.g., an element may begin at a given time, its start can be based upon the *begin* of another element, or upon some event (such as a mouse click). Elements can be defined to be repeated a number of times or for an amount of time. An element's presentation can be given a certain duration or its end can be specified in relation to other events.

The following example gives an impression of the options SMIL Timing provides.

```
<par>

  <audio id="song1" src="song1.au" begin="2s" dur="20s" />

</par>
```

The `<audio>` element "song1" is played 2 seconds after the `<par>` time container begins, and is stopped 20 seconds later. The `` element "img1.jpg" is displayed 2 seconds after the replay of the `<audio>` element is started.

Player Technology

The development of SMIL is tightly coupled with the development of streaming technologies for the WWW. It profits a lot from the fact that it has a rather strong backing from industrial providers of player technologies. Especially RealNetWorks (<http://www.realnetworks.com/>) – producer of the well known RealPlayer – has been supporting SMIL in many of its products. Parts of SMIL (e.g. the TimingModule) are currently also supported by MS-InternetExplorer.

Online References:

Synchronized Multimedia Integration Language (SMIL 2.0) W3C Recommendation 7 August 2001: <http://www.w3.org/TR/smil20>

SMIL Homepage: <http://www.w3org/AudioVideo/>

Conclusion

Multimodal markup for life-like characters is one of the key issues in animated character technology. There is a broad awareness in the research community and in industry that standardization is of great importance. Accordingly a number of standardization efforts are currently underway. On the other hand, single applications call for very specific solutions. Thus there is a need for individual (groups of) developers to create their own, application-specific markup, a situation which we also face in the NECA project.

In developing the RRL we are able to draw on existing standardization efforts and build on well-defined cores of XML-based markup languages, especially in the field of speech synthesis and facial animation. On the other hand, experience gained from our efforts to represent expert markup at all levels of representation (from rather abstract representations of scenes to more or less player specific representations which determine the final output of the NECA system) will hopefully feed into future standardization efforts. To foster exchange between the various approaches to multimodal markup for life-like characters we also taking part in the following activities: (1) We are co-organizing the AAMAS'2002 Workshop on "Embodied conversational agents: let's specify and evaluate them!", 16 July 2002, Bologna, Italy. (2) We are organizing an IST Cross-project Concertation Meeting on Representation Formats/Languages, 18 July, Bologna, Italy

References

- [Ekman & Friesen 1975] W.: Ekman P., Friesen W.: Unmasking the Face, Prentice-Hall, London/New York/Englewood Cliffs, NJ, 1975.
- [Ekman & Friesen 1978] Ekman P., Friesen W.: Facial Action Coding System, Consulting Psychologist Press, Palo Alto, CA, 1978.
- [Gustavsson et al. 2002] Gustavsson C., Strindlund L., Wiknertz E.: Verifiacation, Validation and Evaluation of the Virtual Human Markup Language (VHML), Department of Computer Science and Information Science, Linköping University, LiTH-isy-EX-3188-2002, 2002. [available online at <http://www.ep.liu.se/exjobb/isy/2002/3188>]
- [Kalra 1991] Kalra P.K.: An Interactive Multimodal Facial Animation System, PhD dissertation. Departement d'Informatique, Ecole Polytechnique Federale de Lausanne, 1993.
- [Noot&Ruttkay 2000] Noot H., Ruttkay Z.: CharToon 2.0 manual, CWI, Amsterdam, The Netherlands, 2000. [<http://www.cwi.nl/ftp/CWIreports/INS/INS-R0004.ps.Z>]
- [Ortony et al. 1988] Ortony A., Clore G.L., and Collins A. The Cognitive Structure of Emotions, Cambridge University Press, Cambridge, MA, 1988.
- [Ruttkay&Lelievre 2000] Ruttkay Z., Lelievre A.D.F.: CharToon 2.1 extensions: Expression repertoire and lip sync, CWI, Amsterdam, The Netherlands, 2000. [available online at <http://www.cwi.nl/projects/FASE/CharToon/Papers/INS-R0016.pdf>]
- [Tekalp&Ostermann 2000] Tekalp A.M., Ostermann J.: Face and 2-D Mesh Animation in MPEG-4, Signal Processing: Image Communication, 15 (4-5), pp. 387-421,2000. [available online: http://www.cselt.it/leonardo/icjfiles/mpeg-4_si/8-SNHC_visual_paper.htm]
- [Thorisson 1996] Thorisson K.R.: ToonFace: A System for Creating and Animating Cartoon Faces, MIT Media Laboratory, Cambridge, MA, Learning and Common Sense Section Technical Report 96-01, 1996 [<ftp://ftp.media.mit.edu/pub/kris/toonface.pdf>]
- [Tsutsui et al. 2000] T. Tsutsui, S. Saeyor and M. Ishizuka: MPML: A Multimodal Presentation Markup Language with Character Agent Control Functions, Proc.(CD-ROM) WebNet 2000 World Conf. on the WWW and Internet, San Antonio, Texas, USA, (2000.10.30-2000.11.4) [available online at <http://www.miv.t.u-tokyo.ac.jp/papers/santiWebNet2000.pdf>]
- [Wegener-Knudsen et al. 2002] Wegener-Knudsen M., Martin J.-C.and Dybkjaer L. (eds.) (2002). Survey of Multimodal Annotation Schemes and Best Practice. EAGLES/ISLE Natural Interactivity and Multimodality Working Group, 2002. [available online http://www.ilc.pi.cnr.it/EAGLES96/isle/nimmwg_doc/ISLE_D9.1.zip]

[Zong et al. 2000] Zong Y., Dohi H., Prendinger H., Ishizuka M.: Emotion Expression Function in Multimodal Presentation, Advances in Multimodal Interfaces – ICMI2000 (Proc. 3rd Int'l Conf. on Multimodal Interfaces), Beijing, China, pp.57--64, 2000. [available online at <http://www.miv.t.u-tokyo.ac.jp/papers/yzong-beijing-ICMI2000.pdf>]