VHML

Working Draft v0.2 **September 25th 2001**

This version:
http://www.vhml.org/documents/VHML/2001/WD-VHML-20010925/

Latest version:
http://www.vhml.org/documents/VHML/

Previous version:
http://www.vhml.org/documents/VHML/2001/WD-VHML-20010313/

Editors:

Camilla Gustavsson	Simon Beard
Linda Strindlund	Quoc Huynh
Emma Wiknertz	Andrew Marriott
	John Stallo

Document Maintainer:

vhml@vhml.org

Copyright © 2001 Curtin University of Technology, InterFace. All Rights Reserved.
W3C liability, trademark, document use and software licensing rules apply.

Status of this document
This section describes the status of this document at the time of its publication. Other documents may supersede this document. The latest status of this document series is maintained at the Curtin InterFace Website.

This is the 14th September 2001 Working Draft of the Virtual Human Markup Language Specification.

This working draft relies on the following existing languages:

Facial Animation Markup Language, developed by Huynh (2000).
Speech Markup Language, developed by Stallo (2000).
Speech Synthesis Markup Language, http://www.w3.org/TR/speech-synthesis, developed by W3C.

The various sub languages of VHML use and extend these languages.

Abstract

This document describes the Virtual Human Markup Language, VHML. The language is designed to accommodate the various aspects of human computer interaction with regards to facial animation, text to speech production, body animation, dialogue manager interaction, emotional representation plus hyper and multi media information.

It will use existing standards and will describe new languages to accommodate functionality that is not catered for.

The language will be XML / XSL based and will consist of the following sub languages:

- EML Emotion Markup Language
- GML Gesture Markup Language
- SML Speech Markup Language (based on SSML)
- FAML Facial Animation Markup Language
- BAML Body Animation Markup Language
- XHTML eXtensible HyperText Markup Language
- DMML Dialogue Manager Markup Language (based on W3C Dialogue Manager or AIML)

Although general in nature, the intent of this language is to facilitate the natural and realistic interaction of a Talking Head or Virtual Human with a user via a web page or standalone application. One specific intended use can be found in the deliverables of the Interface project, http://www.ist-interface.org/.

Table of contents

1 TERMINOLOGY AND DESIGN CONCEPTS 4
1.1 RENDERING PROCESSES 4
1.2 DOCUMENT GENERATION, APPLICATIONS AND CONTEXTS 6
2 THE LANGUAGE STRUCTURE 7
3 TOP LEVEL 8
3.1 TOP LEVEL ELEMENTS 8
4 EMOTIONAL MARKUP LANGUAGE (EML) 11
4.1 EML DEFAULT ATTRIBUTES 11
4.2 EML ELEMENTS 12
5 GESTURE MARKUP LANGUAGE (GML) 17
5.1 GML DEFAULT ATTRIBUTES 17
5.2 GML ELEMENTS 17
6 SPEECH MARKUP LANGUAGE (SML) 22
6.1 SML DEFAULT ATTRIBUTES 22
6.2 SML ELEMENTS 22
7 FACIAL ANIMATION MARKUP LANGUAGE (FAML) 28
7.1 FAML DEFAULT ATTRIBUTES 28
7.2 FAML ELEMENTS 28
8 BODY ANIMATION MARKUP LANGUAGE (BAML) 37
9 EXTENSIBLE HYPERTEXT MARKUP LANGUAGE (XHTML) 38
9.1 XHTML DEFAULT ATTRIBUTES 38
9.2 XHTML ELEMENTS 38
10 DIALOGUE MANAGER MARKUP LANGUAGE (DMML) 40
REFERENCES 42
APPENDIX A 44

1 Terminology and design concepts

The design and standardization process has adopted the approach of the Speech Synthesis Markup Requirements for Voice Markup Languages published December 23, 1999 by the W3C Voice Browser Working Group.

The following items were the key design criteria.

- *Consistency:* Provide predictable control of rendering output across platforms and across VHML implementations.
- *Generality:* Support rendering output for a wide range of applications with varied graphics capability and visual as well as speech content.
- *Internationalisation:* Enable visual and speech output in a large number of languages within or across documents.
- *Generation and Readability:* Support automatic generation and hand authoring of documents. The documents should be readable by humans.
- *Implementable:* The specification should be implementable with existing, generally available technology and the number of optional features should be minimal.

1.1 Rendering processes

A rendering system that supports the Virtual Human Markup Language will be responsible for rendering a document as visual and spoken output and for using the information contained in the markup to render the document as intended by the author.

Document creation: A text document provided as input to the system may be produced automatically, by human authoring, or through a combination of these forms. The Virtual Human Markup Language defines the form of the document.

Document processing: The following are the nine major processing steps undertaken by a VHML system to convert marked up text input into automatically generated output. The markup language is designed to be sufficiently rich so as to allow control over each of the steps described below so that the document author (human or machine) can control or direct the final rendered output of the Virtual Human.

1. **XML Parse:** An XML parser is used to extract the document tree and content from the incoming text document. The structure, elements and attributes obtained in this step influence each of the following steps.
2. **Culling of un-needed VHML elements:** For example, at this stage any elements that produce audio when the final rendering device/environment does not support audio may be removed. Similarly for other elements. It should be noted that since the timing synchronisation is based upon vocal production, the spoken text might need to be processed regardless of the output device's capabilities.
3. **Structure analysis:** The structure of a document influences the way in which a document should be read. For example, there are common speaking and acting patterns associated with paragraphs.
   - *Markup support:* Various elements defined in the VHML markup language explicitly indicate document structures that affect the visual and spoken output.
   - *Non-markup behaviour:* In documents and parts of documents where these elements are not used, the VHML system is responsible for inferring the structure by automated analysis of the text, often using punctuation and other language-specific data.

4. **Text normalization:** All written languages have special constructs that require a conversion of the written form (orthographic form) into the spoken form. Text normalization is an automated process of the TTS system that performs this conversion. For example, for English, when "$200" appears in a document it may be spoken as "two hundred dollars". Similarly, "1/2" may be spoken as "half", "January second", "February first", "one of two" and so on.
   - *Markup support:* The **`say-as`** element can be used in the input document to explicitly indicate the presence and type of these constructs and to resolve ambiguities. The set of constructs that can be marked includes dates, times, numbers, acronyms, duration and more. The set covers many of the common constructs that require special treatment across a wide number of languages but is not and cannot be a complete set.
   - *Non-markup behaviour:* For text content that is not marked with the **`say-as`** element the TTS system is expected to make a reasonable effort to automatically locate and convert these constructs to a speakable form. Because of inherent ambiguities (such as the "1/2" example above) and because of the wide range of possible constructs in any language, this process may introduce errors in the speech output and may cause different systems to render the same document differently.
5. **Text-to-phoneme conversion:** Once the system has determined the set of words to be spoken it must convert those words to a string of phonemes. A *phoneme* is the basic unit of sound in a language. Each language (and sometimes each national or dialect variant of a language) has a specific phoneme set. For example, most US English dialects have around 45 phonemes. In many languages this conversion is ambiguous since the same written word may have many spoken forms. For example, in English, "read" may be spoken as [ri:d], “I will read the book” or [redd], “I have read the book”.
   Another issue is the handling of words with non-standard spellings or pronunciations. For example, an English TTS system will often have trouble determining how to speak some non-English-origin names, for example "Tlalpachicatl" which has a Mexican/Aztec origin.
   - *Markup support:* The **`phoneme`** element allows a phonemic sequence to be provided for any word or word sequence. This provides the content creator with explicit control over pronunciations. The **`say-as`** element may also be used to indicate that text is a proper name that may allow a TTS system to apply special rules to determine a pronunciation.
   - *Non-markup behaviour:* In the absence of a **`phoneme`** element the TTS system must apply automated capabilities to determine pronunciations. This is typically achieved by looking up words in a pronunciation dictionary and applying rules to determine other pronunciations. Most TTS systems are expert at performing text-to-phoneme conversions so most words of most documents can be handled automatically.
6. **Prosody analysis:** Prosody is the set of features of speech output that includes the pitch (also called intonation or melody), the timing (or rhythm), the pausing, the speaking rate, the emphasis on words and many other features. Producing human-like prosody is important for making speech sound natural and for correctly conveying the meaning of spoken language.
   - *Markup support:* The **`emphasis`**, **`break`**, **`emphasize-syllable`** and **`prosody`** elements may all be used by document creators to guide the TTS system is generating appropriate prosodic features in the speech output.
   - *Non-markup behaviour:* In the absence of these elements, TTS systems are expert (but not perfect) in automatically generating suitable prosody. This is achieved through

analysis of the document structure, sentence syntax, and other information that can be inferred from the text input.

7. **Waveform production:** The phonemes and prosodic information are used by the TTS system in the production of the audio waveform. There are many approaches to this processing step so there may be considerable platform-specific variation.
   - *Markup support:* The TTS markup does not provide explicit controls over the generation of waveforms. The `voice` and `person` elements allow the document creator to request a particular voice or specific voice qualities, for example a young male voice. The `embed` element allows for insertion of recorded audio data into the output stream.
8. **Facial and body animation production:** Timing information will be used to synchronize the spoken text with facial gestures and expressions as well as with body movements and gestures.
9. **Rendering:** Rendering the multiple streams (Audio, Graphics, Hyper and Multi Media) onto the output device(s).

1.2 Document generation, applications and contexts

There are many classes of document creator that will produce marked up documents to be spoken and expressed by a VHML system. Not all document creators (including human and machine) have access to information that can be used in all of the elements or in each of the processing steps described in the previous section. The following are some of the common cases.

The document creator has no access to information to mark up the text. All processing steps in the VHML system must be performed fully automatically on plain text. The document requires only the root element to indicate the content is to be rendered.

When marked text is generated programmatically the creator may have specific knowledge of the structure and/or special text constructs in some or all of the document. For example, an email reader can mark the location of the time and date of receipt of email. Such applications may use elements that affect structure, text normalization, prosody, possibly text-to-phoneme conversion, as well as facial or body gestures to gain the user's attention.

Some document creators make considerable effort to mark as many details of the document to ensure consistent speech quality across platforms and to more precisely specify output qualities. In these cases, the creator may use any or all of the available elements to tightly control the visual or speech output.

The most advanced document creators may skip the higher-level markup (emotions, facial and body animation tags) and produce low-level VHML markup for segments of documents or for entire documents.

It is important that any XML elements that are part of VHML use existing elements specified in existing (de facto) or developing standards (for example such as HTML or SSML). This will aid in minimising learning curves for new developers as well as maximising opportunities for the emigration of legacy data.

2 The language structure

VHML uses a number of sub languages to facilitate the direction of a Virtual Human interacting with a user via a web page or a standalone application. These sub-languages are:

- Emotional Human Markup Language (EML)
- Gesture Markup Language (GML)
- Speech Markup Language (SML)
- Facial Animation Markup Language (FAML)
- Body Animation Markup Language (BAML)
- eXtensible HyperText Markup Language (XHTML)
- Dialogue Management Markup Language (DMML)

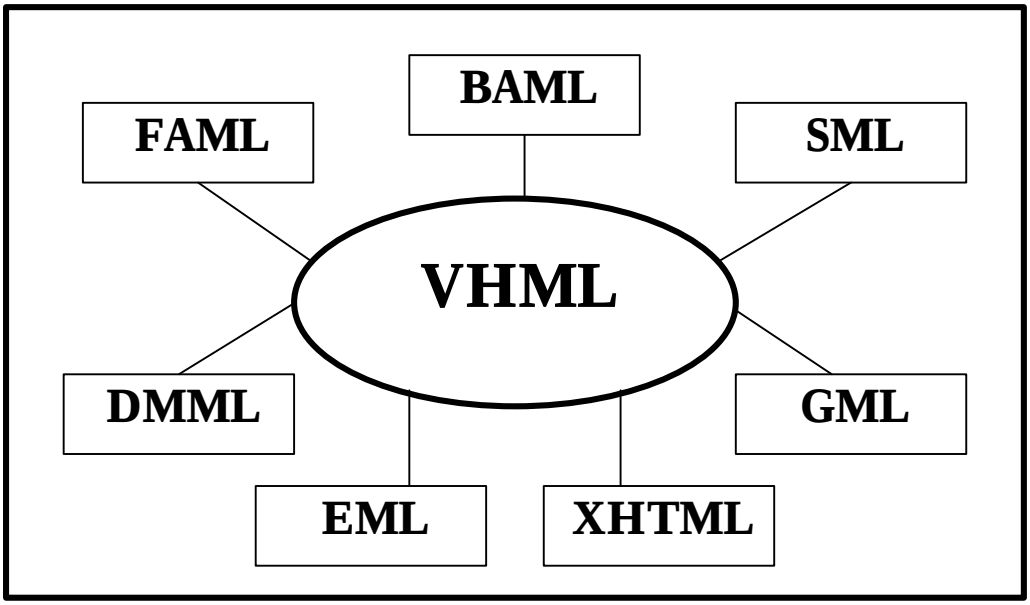

Figure 1. VHML and its sub languages.

In response to a user enquiry, the Virtual Human will have to react in a realistic and humane way using appropriate words, voice, facial and body gestures. For example, a Virtual Human that has to give some bad news to the user may speak in a sad way, with a sorry face and a bowed body stance. In a similar way, a different message may be delivered with a happy voice, a smiley face and with a lively body.

VHML is an XML based language. It uses a DTD in order to describe the rules of the structure of the language. The DTD for VHML is enclosed in appendix A. As with XML elements all VHML elements are case sensitive; therefore all elements must appear in lower case and will otherwise be ignored. When creating a VHML document the first row must contain an XML declaration followed by a DTD specification.

Example:

```
<?xml version="1.0">
<!DOCTYPE vhml SYSTEM "./vhml.dtd">
…
```

3 Top level

The elements at the top level control the structure of the language as well as specify the speaker. An element used to embed foreign files is also placed on this level. No other elements can occur outside these elements.

3.1 Top level elements

The following elements constitute the top level of VHML as it looks today.

`<vhml>`

Description: Root element that encapsulates all other elements.

Attributes:

Name	Description	Values	Default
xml:lang	Indicates the language on the enclosing element.	a language code, following RFC1766	optional

Properties: Can only occur once.
Can contain paragraph, mark and person elements.

Example:

```
<vhml>
…
</vhml>
```

`<person>`

Description: Specifies the person to speak the text, regarding gender, age and category as well as with which emotion it is supposed to speak in general. This emotion will constitute the default emotion for the rest of the element and is used whenever there is no other emotion specified.

Attributes:

Name	Description	Values	Default
mark	Can be used to set an arbitrary mark at a given place in the text, so that an engine can report back to the calling application that it has reached the given location.	a character string that is an identifier for the tag	optional
age	Specifies the preferred age of the voice to speak the contained text.	integer	optional
category	Specifies the preferred age category of the voice to speak the contained text.	child teenager adult elder	optional
gender	Specifies the preferred gender of the voice to speak the contained text.	female male neutral	optional
name	Specifies a platform specific voice name to speak the contained text.	voice-name-list (a space separated list of names	optional

		ordered from top preference down)	
variant	Specifies a preferred variant of another person to speak the contained text.	a character string that starts with the same string as the variant o the person of which it should be a variant	optional
disposition	Specifies the emotion that should be used as default emotion for the contained text.	any of the EML elements	optional

Properties: Can only occur directly under the root element.
Can contain paragraph and mark elements.

Example:

```
<vhml>
      <person age="12" gender="female" disposition="sad"
      variant="fred:1">
            …
      </person>
      <person variant="fred:2">
            …
      </person>
</vhml>
```

<paragraph> = <p>

Description: Element used to divide text into paragraphs. Both the whole word and the abbreviation can be used.

Attributes:

Name	Description	Value	Default
xml:lang	Indicates the language on the enclosing element.	a language code, following RFC1766	optional

Properties: Can only occur directly within a vhml element or a person element.
Can contain plain text as well as all other elements except itself, vhml and person.

Example:

```
<vhml>
      <p> That was the weather for today.</p>
      <p> Regarding the football game yesterday… </p>
</vhml>
```

<mark>

Description: Places a marker into the output stream for asynchronous notification. When the output of the VHML document reaches the mark an event is issued that includes the name attribute. The platform defines the destination of the event. The mark element does not affect the speech or facial animation output process.

Attributes:

Name	Description	Value	Default
mark	Can be used to set an arbitrary mark	a character string	optional

	at a given place in the text, so that an engine can report back to the calling application that it has reached the given location.	that is an identifier for the tag	
name	An identifier for the tag.	a character string	required

Properties: Can occur anywhere in the document under the root element.
An empty element.

Example: **`Go from <mark name="here"/> here, to <mark name="there"/> there.`**

`<embed>`

Description: Gives the ability to embed foreign file types within a VHML document and for them to be processed appropriately.

Attributes:

Name	Description	Value	Default
type	Specifies the type of the embedded file.	audio mml	required
src	Gives the path to the embedded file.	a character string	required

Properties: Can occur anywhere in the document except in vhml and person elements.
An empty element.

Example: **`<embed type="mml" src="song/aaf.mml"/>`**

4 Emotional Markup Language (EML)

The elements in EML will affect the emotion shown by the Virtual Human. These elements will affect the voice, face and body. All emotions will be inherited by SML and FAML.

4.1 EML default attributes

Each element has at least three attributes associated with it.

Name	Description	Value	Default
duration	Represents the time span in seconds or milliseconds that the emotion will persist in the Virtual Human.	#s #ms	required for empty elements and otherwise until closing element
intensity	Represents a percentage value of the maximum intensity of that particular emotion.	0 – 100	100
mark	Can be used to set an arbitrary mark at a given place in the text, so that an engine can report back to the calling application that it has reached the given location.	a character string that is an identifier for the tag	optional
wait	Represents a pause in seconds or milliseconds before continuing with other elements or plain text in the rest of the document.	#s #ms	optional

4.2 EML elements

The following elements constitute EML as it looks today. All the universal emotions are included as well as neutral and two additional emotions. More elements would be profitable and are therefore placed as future work.

<afraid>

Description: Generates a Virtual Human that looks afraid.
Facial animation. The eyebrows are raised and pulled together, the inner eyebrows are bent upward and the eyes are tense and alert.
Speech. The voice is not yet affected by this element.
Body. The body is not yet affected by this element.

Attributes: Default EML attributes.

Properties: Can only occur directly the paragraph element.
Can contain plain text as well as embed and mark elements and all elements on a lower level, i.e. GML, FAML and SML elements.

Example:

```
<afraid intensity="50">
      Do I have to go to the dentist?
</afraid>
```

<angry>

Description: Generates a Virtual Human that looks and sounds angry.
Facial animation. The inner eyebrows are pulled downward and together, the eyes are wide open and the lips are pressed against each other or opened to expose the teeth.
Speech. The speech rate and the pitch of stressed vowels are increased and the average pitch and pitch range are decreased.
Body. The body is not yet affected by this element.

Attributes: Default EML attributes.

Properties: Can only occur directly the paragraph element.
Can contain plain text as well as embed and mark elements and all elements on a lower level, i.e. GML, FAML and SML elements.

Example:

```
<angry>
      You have to clean your room.
</angry>
```

<confused>

Description: Generates a Virtual Human that looks confused.
Facial animation. The eyebrows are bent upwards, the inner eyebrows are having great movement and the corners of the mouth are close together.
Speech. The voice is not yet affected by this element.
Body. The body is not yet affected by this element.

Attributes: Default EML attributes.

Properties: Can only occur directly the paragraph element.
Can contain plain text as well as embed and mark elements and all elements on a lower level, i.e. GML, FAML and SML elements.

Example:

```
<confused intensity="75">
      Where did I put my keys?
</confused>
```

<dazed>

Description: Generates a Virtual Human that looks dazed.
Facial animation. The eyebrows are slightly raised, the eyes opened somewhat wider than normal and the lips are slightly pulled down and outwards.
Speech. The voice is not yet affected by this element.
Body. The body is not yet affected by this element.

Attributes: Default EML attributes.

Properties: Can only occur directly the paragraph element.
Can contain plain text as well as embed and mark elements and all elements on a lower level, i.e. GML, FAML and SML elements.

Example:

```
<dazed duration="10s"/>
      That was a tough sock you gave me.
```

<disgusted>

Description: Generates a Virtual Human that looks disgusted.
Facial animation. The eyebrows and eyelids are relaxed and the upper lid is raised and curled, often asymmetrically.
Speech. The voice is not yet affected by this element.
Body. The body is not yet affected by this element.

Attributes: Default EML attributes.

Properties: Can only occur directly the paragraph element.
Can contain plain text as well as embed and mark elements and all elements on a lower level, i.e. GML, FAML and SML elements.

Example:

```
<disgusted>
      I really hate chocolate cakes.
</disgusted>
```

<happy>

Description: Generates a Virtual Human that looks and sounds happy.
Facial animation. The eyebrows are relaxed, the mouth is open and the mouth corners pulled back towards the ears.
Speech. The speech rate, average pitch and pitch range are increased, so is the duration of the stressed vowels. The changes in pitch between phonemes are eliminated and the amount of pitch fall at the end of an utterance is reduced.
Body. The body is not yet affected by this element.

Attributes: Default EML attributes.

Properties: Can only occur directly the paragraph element.
Can contain plain text as well as embed and mark elements and all elements on a lower level, i.e. GML, FAML and SML elements.

Example:

```
<happy duration=”7s” wait=”2000ms”/>
      It’s my birthday today.
```

<neutral>

Description: Generates a Virtual Human that looks neutral.
Facial animation. All face muscles are relaxed, the eyelids are tangent to iris, lips are in contact, the mouth is closed and the line of the lips is horizontal.
Speech. The voice is not yet affected by this element.
Body. The body is not yet affected by this element.

Attributes: Default EML attributes.

Properties: Can only occur directly the paragraph element.
Can contain plain text as well as embed and mark elements and all elements on a lower level, i.e. GML, FAML and SML elements.

Example:

```
<neutral>
      I’m living in a red house.
</neutral>
```

<sad>

Description: Generates a Virtual Human that looks and sounds sad.
Facial animation. The inner eyebrows are bent upward, the eyes are slightly closed and the mouth is relaxed.
Speech. The speech rate, average pitch and pitch range are decreased. Abrupt changes in pitch between phonemes are eliminated and pauses are added after long words. The pitch for every word before a pause is lowered and all utterances are lowering at the end.
Body. The body is not yet affected by this element.

Attributes: Default EML attributes.

Properties: Can only occur directly the paragraph element.
Can contain plain text as well as embed and mark elements and all elements on a lower level, i.e. GML, FAML and SML elements.

Example:

```
<sad>
      I hurt my knee when I fell in the stairs.
</sad>
```

<surprised>

Description: Generates a Virtual Human that looks surprised.
Facial animation. The eyebrows are raised, the upper eyelids are wide open, the lower relaxed and the jaw is opened.
Speech. The voice is not yet affected by this element.
Body. The body is not yet affected by this element.

Attributes: Default EML attributes.

Properties: Can only occur directly the paragraph element.
Can contain plain text as well as embed and mark elements and all elements on a lower level, i.e. GML, FAML and SML elements.

Example:

```
<surprised>
      I didn’t expect to find that in my lasagne!
</surprised>
```

<default-emotion>

Description: The Virtual Human will get the emotion that is specified in the person element. If a person element does not exist the emotion that is predefined of the application will be used.

Attributes: Default EML attributes.

Properties: Can only occur directly the paragraph element.
Can contain plain text as well as embed and mark elements and all elements on a lower level, i.e. GML, FAML and SML elements.

Example:

```
<default-emotion>
      Now I’m talking in the same way as at the start.
</default-emotion>
```

5 Gesture Markup Language (GML)

The elements in GML will accommodate well-known human gestures. These will affect the voice, face and body of the Virtual Human. All gestures will be inherited by SML and FAML

5.1 GML default attributes

Each element has at least three attributes associated with it.

Name	Description	Value	Default
duration	Represents the time span in seconds or milliseconds that the emotion will persist in the Virtual Human.	#s #ms	required for empty elements and otherwise until closing element
intensity	Represents a percentage value of the maximum intensity of that particular emotion.	0 – 100	100
mark	Can be used to set an arbitrary mark at a given place in the text, so that an engine can report back to the calling application that it has reached the given location.	a character string that is an identifier for the tag	optional
wait	Represents a pause in seconds or milliseconds before continuing with other elements or plain text in the rest of the document.	#s #ms	optional

5.2 GML elements

The following elements constitute GML as it looks today. More elements would be profitable and are therefore placed as future work.

`<agree>`

Description: Generates the Virtual Human to express “yes” or agreement by using gestures.
Facial animation. Animates a nod. It is broken into two sections, the head raise and then the head lower. Only the vertical angle of the head is altered during the element animation, the gaze is still focused forward.
Speech. The speech is not yet affected by this element.
Body. The body is not yet affected by this element.

Attributes: Default GML attributes.

Name	Description	Value	Default
repeat	Specifies how many times the action should occur.	integer	1

Properties: Can occur inside paragraph, EML, emphasis, prosody or voice elements. An empty element.

Example: **`That’s certainly <agree duration=”1000ms”/> right.`**

`<disagree>`

Description: Generates the Virtual Human to express “no” or disagreement by using gestures.

Facial animation. Animates two shakes of the head, which involves first moving to the left, then right, repeated and then returning to the central plane. The element only affects the horizontal displacement of the head and no other facial features are affected.
Speech. The speech is not yet affected by this element.
Body. The body is not yet affected by this element.

Attributes: Default GML attributes.

Name	Description	Value	Default
repeat	Specifies how many times the action should occur.	integer	1

Properties: Can occur inside paragraph, EML, emphasis, prosody or voice elements. An empty element.

Example: **`I <disagree duration="2000ms"/> don't think you are right.`**

`<concentrate>`

Description: Generates a Virtual Human that has a concentrating look.
Facial animation. The eyebrows are lowered and the eyes partly closed.
Speech. The speech is not yet affected by this element.
Body. The body is not yet affected by this element.

Attributes: Default GML attributes.

Properties: Can occur inside paragraph, EML, emphasis, prosody or voice elements. An empty element.

Example: **`<concentrate duration="3s" wait="2s"/> Doing this is really a challenge.`**

`<emphasis>`

Description: Emphasize or accentuate words in the spoken text.
Facial animation. Animates a nod with the eyebrows lowering at the same rate.
Speech. The pitch and duration value are changed.
Body. The body is not yet affected by this element.

Attributes: Default GML attributes.

Name	Description	Value	Default
level	Specifies the strength of emphasis to be applied.	reduced none moderate strong	moderate

Properties: Can occur inside paragraph, EML, emphasis, prosody or voice elements. Can contain plain text as well as embed and mark elements and all elements on a lower level, i.e. GML, FAML and SML elements.

Example: **`I <emphasis level="strong"> will not </emphasis> buy this record, it is scratched.`**

<sigh>

Description: Generates the Virtual Human to express a sigh.
Facial animation. The cheeks are puffed and also the eyebrows, head and mouth are affected.
Speech. The speech is not yet affected by this element.
Body. The body is not yet affected by this element.

Attributes: Default GML attributes.

Name	Description	Value	Default
repeat	Specifies how many times the action should occur.	integer	1

Properties: Can occur inside paragraph, EML, emphasis, prosody or voice elements. An empty element.

Example: **`<sigh duration="2500ms"/> We are still having 2 km left on our walk.`**

<smile>

Description: Generates an expression of a smiling Virtual Human. It is generally used to start sentences and quite often when accentuating positive and cheerful words in a spoken text.
Facial animation. The mouth is widened and the corners pulled back towards the ears.
Speech. The speech is not yet affected by this element.
Body. The body is not yet affected by this element.

Attributes: Default GML attributes.

Properties: Can occur inside paragraph, EML, emphasis, prosody or voice elements. An empty element.

Example: **`<smile duration="2s"/> That was a beautiful dress you've got.`**

Note: A too large intensity value will produce a rather "cheesy" looking grin and can look disconcerting or phony.

<shrug>

Description: Mimics the facial and body expression "I don't know".
Facial animation. The head tilting back, the corners of the mouth pulled downward and the inner eyebrow tilted upwards and squeezed together.
Speech. The speech is not yet affected by this element.
Body. The body is not yet affected by this element.

Attributes: Default EML attributes.

Name	Description	Value	Default
repeat	Specifies how many times the action should occur.	integer	1

Properties: Can occur inside paragraph, EML, emphasis, prosody or voice elements. An empty element.

Example: **<shrug duration=”5000”/> I neither know nor care.**

---

6 Speech Markup Language (SML)

The elements in SML affect the voice of the Virtual Human, the face and body will not be affected. The emotions will be inherited from EML and the gestures from GML.

6.1 SML default attributes

Each element has at least one attribute associated with it.

Name	Description	Value	Default
mark	Can be used to set an arbitrary mark at a given place in the text, so that an engine can report back to the calling application that it has reached the given location.	a character string that is an identifier for the tag	optional

6.2 SML elements

The following elements constitute SML as it looks today. More elements would be profitable and are therefore placed as future work.

`<break>`

Description: Controls the pausing or other prosodic boundaries between words. If the text is not marked up with the element break the speech synthesizer is expected to automatically determine a break based on the linguistic context, for example before starting a new sentence.

Attributes:

Name	Description	Value	Default
size	Specifies the duration of the break.	none small medium large	medium
smooth	Specifies if the last phoneme before the break has to be lengthened slightly.	yes no	yes
time	Specifies the duration of the break in seconds or milliseconds.	#s #ms	optional

Properties: Can occur inside paragraph, EML, emphasis, prosody or voice elements. An empty element.

Example: **`Well <break size="large"/>, I reckon this is a good idea.`**

`<emphasize-syllable> = <emphasise-syllable>`

Description: Emphasizes a syllable within a word. Both spellings of the tag can be used.

Attributes:

Name	Description	Value	Default
affect	Specifies how to emphasize the phoneme.	pitch duration both	pitch

level	Specifies the strength of the emphasis.	reduced none moderate strong	moderate
target	Specifies which phoneme in the text will be emphasized.	a character string representing a phoneme symbol, using MPRA phoneme set	optional

Properties: Can occur inside paragraph, EML, emphasis, prosody or voice elements. Can only contain plain text.

Example:

```
I'm so <emphasize-syllable affect="duration"
level="strong" target="o"> sorry. </emphasize-syllable>
```

<phoneme>

Description: Provides a phonetic pronunciation for the contained text.

Attributes:

Name	Description	Value	Default
alphabet	Specifies which phonetic alphabet that should be used.	ipa worldbet xsampa	optional
ph	Specifies the phoneme string.	a character string	required

Properties: Can occur inside paragraph, EML, emphasis, prosody or voice elements. The element may be empty, but it is recommended that the element contain human readable text.

Example:

```
<phoneme alphabet="ipa" ph="t&#x252;m&#251;to&#x28A;>
tomato </phoneme>
```

<prosody>

Description: Controls the prosody of the contained text.

Attributes:

Name	Description	Value	Default
contour	Specifies the pitch contour for the contained text, with a percentage value of the period of the text (values outside the interval 0% to 100% are ignored) and a pitch, see the pitch attribute for values.	(interval, target), one or many pairs.	optional
duration	Specifies the desired time in seconds or milliseconds take to read the content of the element.	#s #ms	optional
pitch	Specifies the baseline pitch for the contained text.	a numeric relative change low medium high default	default
range	Specifies the pitch range for the contained text.	a numeric relative change low	default

		medium high default	
rate	Specifies the speaking rate for the contained text.	a numeric relative change slow medium fast default	default
volume	Specifies the volume of the contained text.	a numeric relative change silent soft medium loud default	default

Properties: Can occur inside paragraph, EML, emphasis, prosody or voice elements. Can contain plain text as well as embed and mark elements and all elements on a lower level, i.e. GML, FAML and SML elements.

Example:

```
<prosody contour="(0%,20)(10%,+30%)(40%,+10)> Good morning </prosody>

<prosody rate="high" volume="high"> I am talking very fast and very loud. </prosody>
```

Notes: The default value of all the attributes is no change within the element compared to outside the element.
The duration attribute takes precedence over the rate attribute.
The contour attribute takes precedence over the pitch and range attributes.

<say-as>

Description: Controls the pronunciation of the contained text.

Attributes:

Name	Description	Value	Default
type	Specifies the contained text construct. The format is a text type optionally followed by a colon and a format.	acronym number (ordinal, digits) date (dmy, mdy, ymd, ym, my, md, y, m, d) time (hms, hm, h) duration (hms, hm, ms, h, m, s) currency measure telephone name net (email, uri) address	required
sub	Specifies the pronunciation of the contained text.	a character string specifying the string that should be spoken.	optional

Properties: Can occur inside paragraph, EML, emphasis, prosody or voice elements. Can only contain plain text.

Example:

```
<say-as type="date:ymd"> 2001-09-06 </say-as>

<say-as sub="World Wide Consortium"> W3C </say-as>
```

<voice>

Description: Specifies the speaking voice of the contained text.

Attributes:

Name	Description	Value	Default
age	Specifies the preferred age of the voice to speak the contained text.	integer	optional
category	Specifies the preferred age category of the voice to speak the contained text.	child teenager adult elder	optional
gender	Specifies the preferred gender of the voice to speak the contained text.	female male neutral	optional
name	Specifies a platform specific voice name to speak the contained text.	voice-name-list (a space separated list of names ordered from top preference down)	optional
variant	Specifies a preferred variant of the other voice characteristics to speak the contained text.	integer	optional

Properties: Can occur inside paragraph, EML, emphasis, prosody or voice elements. Can contain plain text as well as embed and mark elements and all elements on a lower level, i.e. GML, FAML and SML elements.

Example:

```
<voice gender="female" category="child">
   I am a little girl.
   <voice variant="2"> I'm also a little girl. </voice>
</voice>
```

Notes: The age attribute takes preference over the category attribute.
When there is not a voice available that exactly matches the attributes specified in the document, the voice selection algorithm may be platform specific.
Voice attributes are inherited down a tree structure.

```
<voice gender="female"> Any female voice.
   <voice category="child"> Any female child voice.
   </voice>
</voice>
```

<angry> Inherited from EML.

<confused> Inherited from EML.

<dazed> Inherited from EML.

<disgusted>	Inherited from EML.
<afraid>	Inherited from EML.
<happy>	Inherited from EML.
<neutral>	Inherited from EML.
<sad>	Inherited from EML.
<surprised>	Inherited from EML.
<default-emotion>	Inherited from EML.

<agree>	Inherited from GML.
<disagree>	Inherited from GML.
<concentrate>	Inherited from GML.
<smile>	Inherited from GML.
<shrug>	Inherited from GML.
<sigh>	Inherited from GML.

---

7 Facial Animation Markup Language (FAML)

The elements in FAML affect the facial animation performed by the Virtual Human. These elements will only make changes to the face. The voice and body will not be affected.

The emotions will be inherited from EML and the gestures from GML.

7.1 FAML default attributes

Each element has at least three attributes associated with it.

Name	Description	Value	Default
duration	Represents the time span in seconds or milliseconds that the emotion will persist in the Virtual Human.	#s #ms	required
intensity	Represents a percentage value of the maximum intensity of that particular emotion.	0 - 100	100
mark	Can be used to set an arbitrary mark at a given place in the text, so that an engine can report back to the calling application that it has reached the given location.	a character string that is an identifier for the tag	optional
wait	Represents a pause in seconds or milliseconds before continuing with other elements or plain text in the rest of the document.	#s #ms	optional

7.2 FAML elements

The following elements constitute FAML as it looks today. More elements would be profitable and are therefore placed as future work.

All combinations of the directional elements allow the head to have full range of orientation. A combination of the <look-left> and <look-up> elements will enable to look at the top left in the animation sequence, whilst <look-right> <look-down> will enable the head to look at the bottom right.

`<look-left>`

Description: Turns both the eyes and head to look left. The eyes and head move at the same rate.

Attributes: Default FAML attributes.

Properties: Can occur inside paragraph, EML, emphasis, prosody or voice elements. An empty element.

Example: **`<look-left duration="1500ms"/> Cheese to the left of me.`**

`<look-right>`

Description: Turns both the eyes and head to look right. The eyes and head move at the same rate.

Attributes: Default FAML attributes.

Properties: Can occur inside paragraph, EML, emphasis, prosody or voice elements. An empty element.

Example: **<look-right duration="1s"/> Cheese to the right of me.**

<look-up>

Description: Turns both the eyes and head to look up. The eyes and head move at the same rate.

Attributes: Default FAML attributes.

Properties: Can occur inside paragraph, EML, emphasis, prosody or voice elements. An empty element.

Example: **<look-up duration="5500ms" intensity="85"/> Dear God, is there no escaping this smelly cheese?**

<look-down>

Description: Turns both the eyes and head to look down. The eyes and head move at the same rate.

Attributes: Default FAML attributes.

Properties: Can occur inside paragraph, EML, emphasis, prosody or voice elements. An empty element.

Example: **<look-left duration="2s"/> Perhaps it is just my feet**

The eye directional elements allow four independent directions for eye movement. This entails movement in the vertical and horizontal planes. A combination of the <eyes-left> and <eyes-up> elements will enable to look at the top left in the animation sequence, whilst <eyes-right> <eyes-down> will enable to look at the bottom right.

The eyes cannot be animated independently of each other.

<eyes-left>

Description: The eyes turn left, whilst the head remains in its position.

Attributes: Default FAML attributes.

Properties: Can occur inside paragraph, EML, emphasis, prosody or voice elements. An empty element.

Example: **<eyes-left duration="1000ms" intensity="30"/> There is the door, please use it.**

<eyes-right>

Description: The eyes turn right, whilst the head remains in its postilion.

Attributes: Default FAML attributes.

Properties: Can occur inside paragraph, EML, prosody or voice elements.
An empty element.

Example: **<eyes-right duration="6s"/> A fly flew into my eye, can you see it?**

<eyes-up>

Description: The eyes turn upward, whilst the head remains in its position.

Attributes: Default FAML attributes.

Properties: Can occur inside paragraph, EML, emphasis, prosody or voice elements.
An empty element.

Example: **<eyes-up duration="4s"/> You are just being foolish.**

<eyes-down>

Description: The eyes turn downward, whilst the head remains in its position.

Attributes: Default FAML attributes.

Properties: Can occur inside paragraph, EML, emphasis, prosody or voice elements.
An empty element.

Example: **<eyes-down duration="3300ms"/ intensity="50"> Sorry for breaking your car.**

The animation of the head movement can be broken down into three parts. The first affects the rotational angle of the head the horizontal field, <head-left> and <head-right>. The second affects the elevation and depression of the head in the vertical field, <head-up> and <head-down>.The last affects the axial angle, <head-roll>. The combination of these three factors allows full directional movement for the animation of the Talking Head.

<head-left>

Description: The head turns left, whilst the eyes remain in its position.

Attributes: Default FAML attributes.

Properties: Can occur inside paragraph, EML, emphasis, prosody or voice elements.
An empty element.

Example: **<head-left duration="1000ms" intensity="40"/> Do I have ice creme on my right cheek?**

<head-right>

Description: The head turns right, whilst the eyes remain in its position.

Attributes: Default FAML attributes.

Properties: Can occur inside paragraph, EML, emphasis, prosody or voice elements. An empty element.

Example: **<head-right duration="15s" intensity="40"/> What about my left cheek?**

<head-up>

Description: The head turns upward, whilst the eyes remain in its position.

Attributes: Default FAML attributes.

Properties: Can occur inside paragraph, EML, emphasis, prosody or voice elements. An empty element.

Example: **<head-up duration="2s"/> I'm a bit posh today.**

<head-down>

Description: The head turns downward, whilst the eyes remain in its position.

Attributes: Default FAML attributes.

Properties: Can occur inside paragraph, EML, emphasis, prosody or voice elements. An empty element.

Example: **<head-down duration="6s"/> Sorry, I'm ashamed of what I did.**

<head-roll-left>

Description: Animates a roll of the head to the left in the axial plane. This is essential for adding realism to the Virtual Human and is often used in conjunction with other elements, such as agree and other head movements.

Attributes: Default FAML attributes.

Properties: Can occur inside paragraph, EML, emphasis, prosody or voice elements. An empty element.

Example: **<head-roll-left duration="5s"/> Way of wonder.**

<head-roll-right>

Description: Animates a roll of the head to the right in the axial plane. This is essential for adding realism to the Virtual Human and is often used in conjunction with other elements, such as agree and other head movements.

Attributes: Default FAML attributes.

Properties: Can occur inside paragraph, EML, emphasis, prosody or voice elements. An empty element.

Example: **<head-roll-right duration="1500ms" wait="1s"/> Oh, what a quite dog you've got.**

<eyebrow-up>

Description: Vertical movement upwards with the whole eyebrow. Eyebrow movements are especially used to accentuate words or phrases.

Attributes: Default FAML attributes.

Name	Description	Value	Default
which	Specifies which eyebrow to move.	both left right	both

Properties: Can occur inside paragraph, EML, emphasis, prosody or voice elements. An empty element.

Example: **<eyebrow-up duration="3s" which="right"/> I'm sceptical to what you say.**

<eyebrow-down>

Description: Vertical movement downwards with the whole eyebrow. Eyebrow movements are especially used to accentuate words or phrases.

Attributes: Default FAML attributes.

Name	Description	Value	Default
which	Specifies which eyebrow to move.	both left right	both

Properties: Can occur inside paragraph, EML, emphasis, prosody or voice elements. An empty element.

Example: **<eyebrow-down duration="2400ms"/> I'm really angry with you.**

<eye-blink>

Description: Animates a blink with both eyes. Both the upper and lower eyelids are affected. The intensity value specifies how much of the eyes that should be closed.

Attributes: Default FAML attributes.

Name	Description	Value	Default
repeat	Specifies how many times the action should occur.	integer	1

Properties: Can occur inside paragraph, EML, emphasis, prosody or voice elements. An empty element.

Example: **<eye-blink duration="40ms" repeat="2"/> What a surprise!**

<wink>

Description: Animates a wink of one eye. The wink is not just the blinking of one eye, but the head is affected as well as the outer part of the eyebrow and

cheek. The combination of these animated features add to the realism of the wink itself.

Attributes: Default FAML attributes.

Name	Description	Value	Default
which	Specifies which side to wink.	left right	left
repeat	Specifies how many times the action should occur.	integer	1

Properties: Can occur inside paragraph, EML, emphasis, prosody or voice elements. An empty element.

Example:

```
Nudge, nudge <wink duration="500ms" which="right"/> wink,
<wink duration="2000ms" which="right"/> wink.
```

<open-jaw>

Description: Opens the jaw on a Virtual Human.

Attributes: Default FAML attributes.

Properties: Can occur inside paragraph, EML, emphasis, prosody or voice elements. An empty element.

Example:

```
<open-jaw duration="3s"/> I'm really tired today
<close-jaw duration="2s"/>
```

<close-jaw>

Description: Closes the jaw on a Virtual Human.

Attributes: Default FAML attributes.

Properties: Can occur inside paragraph, EML, prosody or voice elements. An empty element.

Example:

```
<open-jaw duration="3s"/> I think I'm falling asleep.
<close-jaw duration="2s"/>
```

<angry> Inherited from EML.

<confused> Inherited from EML.

<dazed> Inherited from EML.

<disgusted> Inherited from EML.

<afraid> Inherited from EML.

<happy> Inherited from EML.

<neutral> Inherited from EML.

<sad> Inherited from EML.

<surprised> Inherited from EML.

<default-emotion> Inherited from EML.

<agree>	Inherited from GML.
<concentrate>	Inherited from GML.
<disagree>	Inherited from GML.
<smile>	Inherited from GML.
<shrug>	Inherited from GML.
<sigh>	Inherited from GML.

8 Body Animation Markup Language (BAML)

9 eXtensible HyperText Markup Language (XHTML)

The elements in XHTML affect the output text from the application. Only a very limited subset of the actual XH TML is used in VHML.

9.1 XHTML default attributes

Each element has some attribute associated with it.

Name	Description	Value	Default
accesskey	Assigns an access key to the element.	a single character	optional
shape	Specifies the shape of a region.	default rect circle poly	optional
coords	Specifies the position and shape on the screen.	coordinates in percentage separated by commas	optional
tabindex	Specifies the position of the current element in the tabbing order for the current document.	0 – 32 767	optional
onfocus	Occurs when an element receives focus either by pointing device or by tapping navigation.	script data that can be the content of the script element and the value of intrinsic event attributes	optional
onblur	Occurs when an element loses focus either by pointing device or by tapping navigation.	script data that can be the content of the script element and the value of intrinsic event attributes	optional

9.2 XHTML elements

The following element constitutes XHTML as it looks today. More elements are placed as future work.

`<anchor> = <a>`

Description: Inserts an anchor in the output text.

Attributes:

Name	Description	Value	Default
charset	Specifies the character encoding of the resource designated by the link.	a space separated list of character encodings	optional
href	Specifies the location of a web resource, thus defining a link between the current element and the destination anchor.	a URI	optional
hreflang	Specifies the base language of the	a language code,	optional

	resource .	following RFC1766	
name	Names the current anchor so that it may be the destination of another link.	a character string	optional
rel	Describes the relation from the current document to the anchor.	a space separated list of link types	optional
rev	Describe a reverse link from the anchor to the current document.	a space separated list of link types	optional
type	Gives a hint as to the content type of the content available at the link target address.	a content type, following RFC2045 and RFC2046	optional

Properties: Can occur anywhere in the document.
Can only contain plain text.

Example:

```
Please look and find out on
<a href="http://www.vhml.org"> the VHML webpage</a>.
```

---

10 Dialogue Manager Markup Language (DMML)

OK Andrew, here is where you are going to fill in what you think should be there.

References

Bradner, S. (1997), *Key words for use in RFCs to Indicate Requirement Levels*. Available: http://www.normos.org/ietf/rfc/rfc2119.txt [2001, September 12].

Java Speech Markup Language. Available at http://java.sun.com/products/java-media/speech/forDevelopers/JSML/index.html [2001, September 12].

Pelachaud, C and Prevost, S (1995), *Talking heads: Physical, linguistic and cognitive issue in facial animation.* Course Notes for Computer Graphics International '95.

Sable V1.0. Available: http://www.research.att.com/~rws/Sable.v1_0.htm [2001, September 12].

Speech Synthesis Markup Language Specification. Available: http://www.w3.org/TR/speech-synthesis, [2001, September 13].

Speech 2001. Available: http://www.microsoft.com/speech/, [2001, September 14].

Sproat, R. (1998), *The Proper Relation between SABLE and Aural Cascaded Style Sheets*. Available: http://www.bell-labs.com/project/tts/csssable.html, [2001, September 13].

Sproat, R., Hunt, A., Ostendorf, M., Taylor, P., Black, A., Lenzo, K. & Edgington M. (1998), *SABLE: A Standard for TTS Markup*. Available: http://www.research.att.com/~rws/SABPAP/sabpap.htm, [2001, September 13].

TAGS AND ATTRIBUTES. Available: http://www.research.att.com/~rws/SABPAP/node2.htm, [2001, September 13].

Voice eXtensible Markup Language (VocieXML) version 1.0. Available: http://www.w3.org/TR/2000/NOTE-voicexml-20000505/, [2001, September 13].

VoiceXML Forum. Available: http://www.voicexml.org/, [2001, September 14].

Acknowledgements

Thanks to Ania Wojdel and Michele Cannella for their contribution with opinions about and proposed solutions to the structure of VHML.

Appendix A

```
<!--
#####################################################################
# Virtual Human Markup Language (VHML) DTD, version 0.4.            #
#                                                                   #
# Usage:                                                            #
# <!DOCTYPE vhml SYSTEM "./vhml.dtd">                               #
#                                                                   #
# Author: Camilla Gustavsson, c.gustavsson@home.se                  #
#         Linda Strindlund, linda.strindlund@home.se          `     #
#         Emma Wiknertz, wiknertz@home.se                           #
#                                                                   #
# Information about the VHML can be found at http://www.vhml.org    #
#                                                                   #
# Date: 14 September, 2001.                                         #
#                                                                   #
#####################################################################
-->

<!--
#########################################
# Some entities for an abstracter view #
#########################################
-->

<!-- COMMENT:
New emotions are added here and specified below.
-->
<!ENTITY % EML     "afraid |
                   angry |
                   confused |
                   dazed |
                   disgusted |
                   happy |
                   neutral |
                   sad |
                   surprised |
                   default-emotion">

<!ENTITY % Emotion "( %EML; )">

<!-- COMMENT:
New gestures are added here and specified below.
-->
<!ENTITY % GML     "agree |
                   disagree |
                   concentrate |
                   emphasis |
                   sigh |
                   smile |
                   shrug">
<!-- COMMENT:
New FAML elements are added here and specified below.
-->
<!ENTITY % FAML    "look-left |
                   look-right |
                   look-up |
                   look-down |
```

```
                  eyes-left |
                  eyes-right |
                  eyes-up |
                  eyes-down |
                  head-left |
                  head-right |
                  head-up |
                  head-down |
                  head-roll-left |
                  head-roll-right |
                  eyebrow-up |
                  eyebrow-down |
                  eye-blink |
                  wink |
                  open-jaw |
                  close-jaw">

<!-- COMMENT:
New SML elements are added here and specified below.
-->
<!-- COMMENT:
These elements are taken from SSML, Speech Synthesis Markup Language.
Some more attributes to the elements are added.
http://www.w3.org/TR/speech-synthesis
-->
<!ENTITY % SML    "break |
                  emphasize-syllable |
                  emphasise-syllable |
                  phoneme |
                  prosody |
                  say-as |
                  voice">

<!ENTITY % XHTML  "a |
                  anchor”>

<!ENTITY % allowed-on-lower-level
            "(#PCDATA | mark | embed | %GML; | %FAML; | %SML; |
            %XHTML;)*">

<!ENTITY % sourcepath "CDATA">

<!ENTITY % integer "CDATA">

<!ENTITY % secs-or-msecs "CDATA">

<!ENTITY % id "CDATA">

<!ENTITY % substitute-string "CDATA">

<!ENTITY % phoneme-string "CDATA">

<!ENTITY % contour-format "CDATA"> <!-- from SSML -->

<!-- COMMENT:
Can be a relative change or one of
low, medium, high or default.
-->
<!ENTITY % pitchvalues "CDATA">

<!-- COMMENT:
```

```
Can be a relative change or one of
low, medium, high or default.
-->
<!ENTITY % rangevalues "CDATA">

<!-- COMMENT:
Can be a relative change or one of
slow, medium, fast or default.
-->
<!ENTITY % ratevalues "CDATA">

<!-- COMMENT:
Can be a relative change or one of
silent, soft, medium, loud or default.
-->
<!ENTITY % volumevalues "CDATA">

<!ENTITY % voice-name-list "CDATA"> <!-- from SSML -->

<!ENTITY % link-type-list "CDATA">

<!ENTITY % character-list "CDATA">

<!ENTITY % uri "CDATA">

<!ENTITY % coordinate-list "CDATA">

<!ENTITY % script "CDATA">

<!ENTITY % say-as-types
            "(acronym | number | number:ordinal | number:digits |
            date | date:dmy | date:mdy | date:ymd | date:ym |
            date:my | date:md | date:y | date:m | date:d | time |
            time:hms | time:hm | time:h | duration | duration:hms |
            duration:hm | duration:ms | duration:h | duration:m |
            duration:s | currency | measure | telephone | name |
            net | net:email | net:uri | address )">
            <!-- from SSML -->

<!ENTITY % default-EML-attributes
            "duration %secs-or-msecs; #IMPLIED
            intensity %integer; '100'
            mark %id; #IMPLIED
            wait %secs-or-msecs #IMPLIED">

<!ENTITY % default-GML-attributes
            "duration %secs-or-msecs; #REQUIRED
            intensity %integer; '100'
            mark %id; #IMPLIED
            wait %secs-or-msecs #IMPLIED">

<!ENTITY % default-FAML-attributes
            "%default-GML-attributes;">

<!ENTITY % default-XHTML-attributes
            "accesskey %id; #IMPLIED
            coords %coordinate-list; #IMPLIED
            onblur %script; #IMPLIED
            onfocus %script; #IMPLIED
            shape (default | rect | circle | poly) #IMPLIED
```

```
            tabindex %integer; #IMPLIED"> <!-- The tabindex must be
            between 0 and 32,767 -->

<!--
####################
# Elements in VHML #
####################
-->

<!ELEMENT vhml (paragraph | p | person | mark)+>
<!ATTLIST vhml
      xml:lang NMTOKEN #IMPLIED>

<!ELEMENT person (paragraph | p | mark)*>
<!ATTLIST person
      mark %id; #IMPLIED
      age %integer; #IMPLIED
      category (child | teenager | adult | elder) #IMPLIED
      gender (female | male | neutral) #IMPLIED
      name %voice-name-list; #IMPLIED
      variant %integer; #IMPLIED
      disposition %Emotion; #IMPLIED>

<!ELEMENT paragraph (#PCDATA | mark | embed | %EML; | %GML; | %FAML;
                     | %SML; | %XHTML;)*>
<!ATTLIST paragraph
      xml:lang NMTOKEN #IMPLIED>

<!ELEMENT p (#PCDATA | mark | embed | %EML; | %GML; | %FAML; |
            %SML; | %XHTML;)*>
<!ATTLIST p
      xml:lang NMTOKEN #IMPLIED>

<!ELEMENT mark EMPTY>
<!ATTLIST mark
      mark %id; #IMPLIED
      name CDATA #REQUIRED>

<!ELEMENT embed EMPTY>
<!ATTLIST embed
      type (audio | mml) #REQUIRED
      src %sourcepath; #REQUIRED>

<!--
####################
# Elements in EML #
####################
-->

<!ELEMENT afraid %allowed-on-lower-level;>
<!ATTLIST afraid %default-EML-attributes;>

<!ELEMENT angry %allowed-on-lower-level;>
<!ATTLIST angry %default-EML-attributes;>

<!ELEMENT confused %allowed-on-lower-level;>
<!ATTLIST confused %default-EML-attributes;>

<!ELEMENT dazed %allowed-on-lower-level;>
<!ATTLIST dazed %default-EML-attributes;>
```

```
<!ELEMENT disgusted %allowed-on-lower-level;>
<!ATTLIST disgusted %default-EML-attributes;>

<!ELEMENT happy %allowed-on-lower-level;>
<!ATTLIST happy %default-EML-attributes;>

<!ELEMENT neutral %allowed-on-lower-level;>
<!ATTLIST neutral %default-EML-attributes;>

<!ELEMENT sad %allowed-on-lower-level;>
<!ATTLIST sad %default-EML-attributes;>

<!ELEMENT surprised %allowed-on-lower-level;>
<!ATTLIST surprised %default-EML-attributes;>

<!--COMMENT:
This is for the default emotion in the person element if there is
one.
Otherwise the system default emotion will be used.
-->
<!ELEMENT default-emotion %allowed-on-lower-level;>
<!ATTLIST default-emotion %default-EML-attributes;>

<!--
####################
# Elements in GML #
####################
-->

<!ELEMENT agree EMPTY>
<!ATTLIST agree %default-GML-attributes;
            repeat %integer '1'>

<!ELEMENT disagree EMPTY>
<!ATTLIST disagree %default-GML-attributes;
            repeat %integer '1'>

<!ELEMENT concentrate EMPTY>
<!ATTLIST concentrate %default-GML-attributes;>

<!ELEMENT emphasis %allowed-on-lower-level;>
<!ATTLIST emphasis
            duration %secs-or-msecs; #IMPLIED
            intensity %integer; '100'
            mark %id; #IMPLIED
            level (reduced | none | moderate | strong) 'moderate'>

<!ELEMENT sigh EMPTY>
<!ATTLIST sigh %default-GML-attributes;
            repeat %integer '1'>

<!ELEMENT smile EMPTY>
<!ATTLIST smile %default-GML-attributes;>

<!ELEMENT shrug EMPTY>
<!ATTLIST shrug %default-GML-attributes;
            repeat %integer '1'>


<!--
##################
```

```
# Element in SML #
##################
-->

<!ELEMENT break EMPTY>
<!ATTLIST break
      mark %id; #IMPLIED
      size (none | small | medium | large) 'medium'
      time %secs-or-msecs; #IMPLIED
      smooth (yes | no) 'yes'>

<!ELEMENT emphasize-syllable (#PCDATA)>
<!ATTLIST emphasize-syllable
      mark %id; #IMPLIED
      target %phoneme-string; #IMPLIED
      level (reduced | none | moderate | strong) 'moderate'
      affect (pitch | duration | both) 'pitch'>

<!ELEMENT emphasise-syllable (#PCDATA)>
<!ATTLIST emphasise-syllable
      mark %id; #IMPLIED
      target %phoneme-string; #IMPLIED
      level (reduced | none | moderate | strong) 'moderate'
      affect (pitch | duration | both) 'pitch'>

<!ELEMENT phoneme (#PCDATA)>
<!ATTLIST phoneme
      mark %id; #IMPLIED
      alphabet (ipa | worldbet | xsampa) #IMPLIED
      ph %phoneme-string; #REQUIRED>

<!ELEMENT prosody %allowed-on-lower-level;>
<!ATTLIST prosody
      mark %id; #IMPLIED
      contour %contour-format; #IMPLIED
      duration %secs-or-msecs; #IMPLIED
      pitch %pitchvalues; 'default'
      range %rangevalues; 'default'
      rate %ratevalues; 'default'
      volume %volumevalues; 'default'>

<!ELEMENT say-as (#PCDATA)>
<!ATTLIST say-as
      mark %id; #IMPLIED
      type %say-as-types; #REQUIRED
      sub %substitute-string; #IMPLIED>

<!ELEMENT voice %allowed-on-lower-level;>
<!ATTLIST voice
      mark %id; #IMPLIED
      age %integer; #IMPLIED
      category (child | teenager | adult | elder) #IMPLIED
      gender (female | male | neutral) #IMPLIED
      name %voice-name-list; #IMPLIED
      variant %integer; #IMPLIED>


<!--
####################
# Elements in FAML #
####################
```

```
-->

<!ELEMENT look-left EMPTY>
<!ATTLIST look-left
            %default-FAML-attributes;>

<!ELEMENT look-right EMPTY>
<!ATTLIST look-right
            %default-FAML-attributes;>

<!ELEMENT look-up EMPTY>
<!ATTLIST look-up
            %default-FAML-attributes;>

<!ELEMENT look-down EMPTY>
<!ATTLIST look-down
            %default-FAML-attributes;>


<!ELEMENT eyes-left EMPTY>
<!ATTLIST eyes-left
            %default-FAML-attributes;>

<!ELEMENT eyes-right EMPTY>
<!ATTLIST eyes-right
            %default-FAML-attributes;>

<!ELEMENT eyes-up EMPTY>
<!ATTLIST eyes-up
            %default-FAML-attributes;>

<!ELEMENT eyes-down EMPTY>
<!ATTLIST eyes-down
            %default-FAML-attributes;>


<!ELEMENT head-left EMPTY>
<!ATTLIST head-left
            %default-FAML-attributes;>

<!ELEMENT head-right EMPTY>
<!ATTLIST head-right
            %default-FAML-attributes;>

<!ELEMENT head-up EMPTY>
<!ATTLIST head-up
            %default-FAML-attributes;>

<!ELEMENT head-down EMPTY>
<!ATTLIST head-down
            %default-FAML-attributes;>


<!ELEMENT head-roll-left EMPTY>
<!ATTLIST head-roll-left
            %default-FAML-attributes;>

<!ELEMENT head-roll-right EMPTY>
<!ATTLIST head-roll-right
            %default-FAML-attributes;>
```

```
<!ELEMENT eyebrow-up EMPTY>
<!ATTLIST eyebrow-up
            %default-FAML-attributes;
            which (both | left | right) 'both'>

<!ELEMENT eyebrow-down EMPTY>
<!ATTLIST eyebrow-down
            %default-FAML-attributes;
            which (both | left | right) 'both'>

<!ELEMENT eye-blink EMPTY>
<!ATTLIST eye-blink
            %default-FAML-attributes;
            repeat %integer '1'>

<!ELEMENT wink EMPTY>
<!ATTLIST wink
            %default-FAML-attributes;
            which (left | right) 'left'
            repeat %integer '1'>

<!ELEMENT open-jaw EMPTY>
<!ATTLIST open-jaw
            %default-FAML-attributes;>

<!ELEMENT close-jaw EMPTY>
<!ATTLIST close-jaw
            %default-FAML-attributes;>

<!--
#####################
# Elements in XHTML #
#####################
-->

<!ELEMENT a (#PCDATA)>
<!ATTLIST a
      %default-XHTML-attributes;
      charset %character-list; #IMPLIED
      href %uri; #IMPLIED
      hreflang NMTOKEN #IMPLIED
      name %id; #IMPLIED
      rel %link-type-list; #IMPLIED
      rev %link-type-list; #IMPLIED
      type NMTOKEN #IMPLIED
      >

<!ELEMENT anchor (#PCDATA)>
<!ATTLIST anchor
      %default-XHTML-attributes;
      charset %character-list; #IMPLIED
      href %uri; #IMPLIED
      hreflang NMTOKEN #IMPLIED
      name %id; #IMPLIED
      rel %link-type-list; #IMPLIED
      rev %link-type-list; #IMPLIED
      type NMTOKEN #IMPLIED
      >
```