Cross-Media and Elastic Time Adaptive Presentations: the Integration of a Talking Head Tool into a Hypermedia Formatter*

Rogério Ferreira Rodrigues[1], Paula Salgado Lucena Rodrigues[1], Bruno Feijó[1], Luiz Velho[2], and Luiz Fernando Gomes Soares[1]

[1] DI – PUC-Rio
Rua Marquês de São Vicente, 225, Rio de Janeiro, RJ, Brazil - 22453-900.
{rogerio, pslucena, bruno, lfgs}@inf.puc-rio.br
[2] IMPA – Inst. de Mat. Pura e Aplicada
Estrada Dona Castorina, 110, Rio de Janeiro, RJ, Brazil - 22460-320.
{lvelho}@impa.br

Abstract. This paper describes the integration of a facial animation tool (*Expressive Talking Heads - ETHs*) with an adaptive hypermedia formatter (*HyperProp formatter*). This formatter is able to adjust document presentations based on the document temporal constraints (e.g. synchronization relationships), the presentation platform parameters (e.g. available bandwidth and devices), and the user profile (e.g. language, accessibility, etc.). This work describes how ETHs augments the capability for creating adaptive hypermedia documents with HyperProp formatter. The paper also presents the adaptation facilities offered by the main hypermedia language (*Nested Context Language - NCL*) HyperProp system works with, and details the implementation extensions of Expressive Talking Heads that turned it an adaptive presentation tool.

1 Introduction

The expressiveness power of a hypermedia presentation system is closely related to its capacity of dealing with different media formats. On the other hand, the decoupling of the presentation management functions (e.g. synchronization control, presentation adaptation, prefetching control, etc.) from the media content exhibition tasks gives more flexibility and extensibility to the system [15]. Therefore, the proposal is to establish an architecture where media players (presentation tools) are modules that can be plugged to the presentation system core element, which is usually named *hypermedia formatter* (or *hypermedia engine*). To accomplish this goal, an opened interface should be specified by the formatter to allow not only the incorporation of external presentation tools, but also to enable these incorporated presentation tools to interact with each other.

* This work was granted by the Brazilian Telecommunications Technological Development Fund (FUNTTEL), through contract 0594/02, and by CNPq.

Following this approach, this paper describes the integration of the HyperProp system formatter [15, 16] with a talking head presentation tool, named Expressive Talking Heads (ETHs) [12]. ETHs is a tool able to present a synthesized speech, synchronized with lip movements and facial expressions (mainly emotions), where the speech and face animation are dynamically generated from an input markup text.

The HyperProp formatter uses an event-driven model to control document presentations. The system can receive document specifications in higher-level authoring languages, like NCL [13] and SMIL [18]. In order to illustrate the adaptation capabilities of HyperProp formatter integrated with ETHs, the paper uses an NCL document example.

Cross-media adaptations [2] are supported by switch elements, allowing, for example, the same text content to be presented by a talking head tool or as a common formatted text. Switch elements can also allow the synthesized speech to be adapted to different language accents. Elastic-time adaptation [1, 9, 11], in the context of this work, permits the synthesized speech to be played with different speeds, aiming at maintaining the voice temporal relationships with other media objects, thus meeting the requirements of the document specifications. All adaptations are guided by context parameters, like user and platform characteristics, and can be static or dynamic. The NCL document example used in the paper illustrates these possible adaptations.

The paper is organized as follows. Sect. 2 describes ETHs and explains how the integration with HyperProp formatter was conducted. Sect. 3 illustrates with a document example the system facilities for developing adaptive hypermedia documents, discussing the adaptation issues in the integrated system. Sect. 4 compares the proposal with related work. Finally, Sect. 5 presents the paper conclusions.

2 Expressive Talking Heads meets HyperProp

2.1 Expressive Talking Heads

The Expressive Talking Heads (ETHs) is a tool that is able to interactively receive markup texts and to generate the animation of a virtual character face speaking the texts [12]. The text markups can set speech idiom accents (e.g. American or British English), voice gender (e.g. male or female), character emotion (e.g. natural, frightened, annoyed, happy), eyes and head positioning, text anchors, among others. The anchor markup, in particular, is a very important feature, since it gives authors the possibility of defining relationships among arbitrary speech segments (spoken by ETHs) and other document media objects, which can eventually be other speech segments. Fig. 1 depicts the ETHs architecture.

ETHs contains a parser component that is responsible for separating the speech content itself (text without markups) from the speech and animation markups. The parser interacts with the tool synthesizer to build the facial animation and lip-sync data structures.

The ETHs parser sends each fragment of marked text to the synthesizer (Festival [17] and MBROLA [5] in the figure) that first creates the speech phonetic description (list of phoneme entries, each one containing the phoneme label, duration and pitch). From this phonetic structure, the parser can identify the phonemes corresponding to the beginning and end of the fragment and thus to assign the beginning and end phonemes for each anchor, emotion, eyes positioning, etc. All these information are stored in the animation data structure. When finishing handling all fragments of marked text, the parser concatenates the several phonetic structures and, together with the speech metadata (idiom, gender, etc.), sends these new data to the synthesizer to have the digitized speech audio generated. Note that it is possible to adapt the speech idiom and the voice gender to a user preference, since the MBROLA synthesizer offers a rich base of distinct voices.

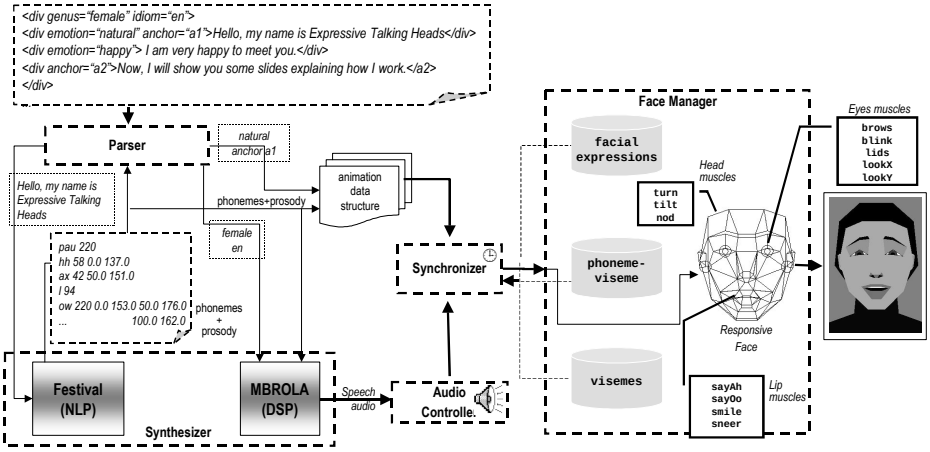

Fig. 1. Overview of the ETHs architecture.

During the synthesis process, the ETHs parser can modify the phonetic structure in order to produce a more realistic speech sound, for example, introducing a random pause-duration between sentences.[1] Moreover, the ETHs tool is designed to permit its user applications to interfere in the synthesis process. This can be used, as explained later, to adjust the duration of the synthesized speech, adapting the hypermedia presentation to context parameters.

The face manager module of ETHs links it to another external subsystem, named Responsive Face [14]. Actually, the ETHs face was inherited from this subsystem, which defines a three-dimensional polygonal mesh, as illustrated in Fig. 1. The face is animated by the application of relax and contract commands over the mesh edges (face muscles). ETHs improves the Responsive Face features adding speech and ap-

[1] As another example, when text markups appear inside a sentence, the created phonetic structure contains one pause phoneme at its beginning and another one at its end. If the phonemes were simply concatenated, the resultant speech sound would have uncomfortable gaps. Therefore, the ETHs parser suppresses these kinds of pause phonemes.

plying to its face the concept of visemes. Viseme is the name given to a mouth configuration for a specific phoneme. The ETHs face manager defines a table containing the phoneme-viseme mapping and also a base of 16 visemes for English phonemes [6], inheriting the 8 emotion facial expressions already defined by Responsive Face. Each base entry stores the values for contracting/relaxing the face corresponding muscles commanding the Responsive Face.

The ETHs synchronizer module is responsible for the fine synchronization between the speech and the facial muscle movements. Parallel to the audio file reproduction, the synchronizer polls the audio controller to check the effective playing instant. Using the phoneme durations, the synchronizer discovers the current phoneme and the current character emotion. Then the synchronizer gets the associated viseme and facial expression muscle contracting/relaxing values, and asks the face manager to apply these values over the Responsive Face. Actually, instead of working just with the current phoneme, the synchronizer uses diphones (two consecutive phonemes interpolated by their durations), since the lip positions for the same phoneme generally changes according to the phoneme context (speech co-articulation aspect).

The synchronizer also monitors the text anchor list in order to find when a speech anchor begins or finishes. These events are notified to components that had register themselves as ETHs anchor observers [8]. The synchronization module also has components to control the head and eyes movements, in order to produce a more natural output.

To facilitate the ETHs use, its services are available through a single facade [8] that hides the internal organization described in this section. The Expressive Talking Heads and the Responsive Face are currently implemented in Java. More details about Expressive Talking Heads project can be found at http://www.telemidia.puc-rio.br/~pslr/eths/.

2.2 Presentation Tool Adapter for Integrating ETHs into HyperProp

To enable the incorporation of new presentation tools (or media players), the HyperProp formatter specifies an integration API. This interface defines the methods that the presentation tools must implement and how they should notify the formatter about presentation event occurrences (user interaction, start/end of content anchor presentation, etc.). Media players that do not implement the required methods, or do not know how to control the HyperProp event state machine [15], should be plugged to the formatter through an adapter component [8]. This was the strategy used to integrate ETHs with the HyperProp formatter.

When initialized, the ETHs adapter receives from the formatter the media object that should be controlled and a descriptor; a set of parameters specifying how the object should be played. Besides the content reference, the media object contains the list of presentation events that should be monitored. These events have associated anchors (fragments of text to be marked up).

Different from the embedded anchor model used by ETHs, the HyperProp formatter allows anchors to be externally defined. This separation aims to allow specifying relationships among hypermedia nodes independent from their content. As a conse-

quence of the different anchor paradigms, the ETHs adapter must analyze the formatter external anchors and then dynamically embed them into the text content, using the ETHs markup definitions (Sect. 2.1). Afterwards the adapter asks the ETHs (through its facade) to perform the synthesis.

Among all its presentation parameters, the descriptor has an attribute specifying the expected duration of the media-object content presentation. The ETHs adapter compares this value with the sum of all phoneme durations and tries to make them equal by adjusting the phoneme durations. If the adapter does not succeed on this task, it reports the fact to the formatter. As mentioned in the previous section, the adapter can alter the phonetic structure since it is an ETHs user application. It does this, basically, modifying the pause phonemes. It is worth mentioning that the adapter may not only adjust the whole content duration, but also adapt each anchor-duration separately.

After all elastic time adaptations, the adapter asks the ETHs for concluding the synthesis. The adapter then registers itself as an ETHs anchor observer and waits for the presentation start command coming from the formatter. When the start command is triggered, the adapter passes the request to the ETHs, which initiates its synchronizer component. The adapter then stays monitoring anchor notifications to convert them into transitions in HyperProp event state machines [15].

3 Adaptive Document Presentation in HyperProp+ETHs

Fig. 2 presents an example of hypermedia document specification using NCL [13]. Due to the lack of space, some document parts were omitted, being replaced with ellipses.[2] The HyperProp formatter has a converter that translates the NCL specification into its execution model [16].

When preparing the document presentation, the HyperProp formatter first analyses the document switches (analogous to SMIL switches [18]) to select, for each of them, the best alternative given by the test rules. However, different from SMIL, the switch test attributes are defined in a presentation rule base (lines 5-18), enabling the same rule to be reused more than once. In the example, the defined rules allow testing the user knowledge level, and the user preference for synthesis and the speech accent. As it can be observed, some rules are grouped defining composite rules. The parameters used in the presentation rules are based on the formatter contextual information (user profile and platform characteristics). In the current implementation, the formatter locally maintains the contextual information, but it is possible to integrate the formatter with third-party context management systems [16]. The formatter also has a dialog interface that enables users editing context parameter values.

```
01 <?xml version="1.0" encoding="UTF-8"?>
02 <ncl ...>
03 <head>
04 <layout>...</layout>
05 <presentationRuleBase>
06  <presentationRule id=”rule01” var=”userLevel” op=”eq” value=”beginner”/>
```

[2] The complete NCL document and a downloadable version of HyperProp formatter can be found at http://www.telemidia.puc-rio.br/products/formatter/.

```
07  <presentationRule id=”rule02” var=”userLevel” op=”eq” value=”expert”/>
08  <presentationRule id=”rule03” var=”synthEnabled” op=”eq” value=”true”/>
09  <presentationRule id=”rule04” var=”synthEnabled” op=”eq” value=”false”/>
10  <compositePresentationRule id=”rule05” op=”and”>
11   <presentationRule id=”rule05a” idref=”rule03”/>
12   <presentationRule id=”rule05b var=”speechAccent” op=”eq” value=”en”/>
13  </compositePresentationRule>
14  <compositePresentationRule id=”rule06” op=”and”>
15   <presentationRule id=”rule06a” idref=”rule03”/>
16   <presentationRule id=”rule06b var=”speechAccent” op=”eq” value=”en-uk”/>
17  </compositePresentationRule>
18 </presentationRuleBase>
19 <costFunctionBase>
20  <costFunction id=”speechCost” xsi:type=”linear”
21    deltaShrink=”15%”  deltaStretch=”15%”  minDurCost=”10”  maxDurCost=”10”/>
22 </costFunctionBase>
23 <descriptorBase>
24  <descriptor id="videoDesc" dur=”30” .../>
25  <descriptor id="subtDesc" nodeRule=”rule03”.../>
26  <descriptorSwitch id="explanationDesc">
27   <bindRule rule=”rule04” component=”textDesc”/>
28   <bindRule rule=”rule05” component=”speechDescEn”/>
29   <bindRule rule=”rule06” component=”speechDescUk”/>
30   <descriptor id="textDesc" .../>
31   <descriptor id="speechDescEn" costFunction=”speechCost”
32     player=”HF_ETHsAdapter”>
33    <param name=”idiom” value=”en”/>
34   </descriptor>
35   <descriptor id="speechDescUk" costFunction=”speechCost”
36     player=”HF_ETHsAdapter”>
37    <param name=”idiom” value=”en-uk”/>
38   </descriptor>
39  </descriptorSwitch>
40 </descriptorBase>
41 </head>
42 <body>
43  <port id=”entryPoint” component=”explanation”>
44  <video id=”video” descriptor=”videoDesc” src=”...”/>
45  <text id=”subtitle” descriptor=”subtDesc” src=”...”/>
46  <switch id=”explanation”>
47   <bindRule rule=”rule01” component=”beginnerExplanation”/>
48   <bindRule rule=”rule02” component=”expertExplanation”/>
49   <text id=”beginnerExplanation” src=”...” descriptor=”explanationDesc”/>
50   <text id=”expertExplanation” src=”...” descriptor=”explanationDesc”/>
51  </switch>
52  <linkBase>...</linkBase>
53 </body>
54 </ncl>
```

Fig. 2. Adaptive NCL document.

The document node switch defined in lines 46-51 specifies two alternatives of text nodes, whose selection depends on the user knowledge level. Besides content alternatives (node switches), NCL (and the HyperProp formatter) allows authors defining alternatives of presentation characteristics for the same content, denoted as descriptor switches (lines 26-39). This possibility makes the ETHs tool very useful. In the example, the same text explanation has three presentation alternatives: if the user does not enable the speech synthesis feature, the text will be present in its original format using the *textDesc* descriptor; if the user agrees on having speech synthesis and the user speech accent preference is for American English, the text node will be presented using the *speechDescEn* descriptor. Otherwise, the text will be synthesized and presented with a British accent. As it is depicted in Fig. 2, node and descriptor switches can be combined, automatically generating a larger amount of alternatives. In the

example, the node switch combined with the nested descriptor switches gives 6 different alternatives for presenting the explanation to the user.

Still in the switch adaptation phase, the formatter looks for rules directly associated with descriptors, as in the subtitle descriptor in line 25. In this example, since the subtitle text node (line 45) is associated with the *subtDesc* descriptor, the text node will be enabled during the presentation only if the user chooses to have the text synthesis (*rule03*), otherwise the explanation will appear as a text and the subtitle becomes unnecessary (disabled).

If the formatter encounters a rule that cannot be resolved at presentation compile time, the formatter maintains the switch information in its execution plan [16], and delays the selection to be done on-the-fly. Similarly, if a context parameter value changes after initiating the presentation, the formatter re-evaluates the rules and, therefore, the node and descriptor switches.

After this first adaptation phase, the HyperProp formatter runs the cross-media adaptation mechanisms. The formatter looks for nodes with presentation characteristics that specify a content transformation. In Fig. 2, the descriptor *player* attribute in lines 32 and 36 gives this content transformation information, specifying ETHs as the presentation tool for node contents that are associated with this descriptor. If this attribute was not defined, the formatter would instantiate the default presentation tool for showing the node content, as occurs when the *textDesc* descriptor is selected.

In the third adaptation phase, the HyperProp formatter runs its elastic time algorithm [1]. In NCL (and also in the HyperProp execution model), the spatio-temporal synchronization among media objects is specified through links, grouped into the document linkbases (line 52) [13]. The formatter uses object durations and link specifications to build the document time chains [16]. NCL and the HyperProp formatter also enable object durations to be flexibly specified. In the example, there is an element (lines 20-22) specifying that durations using the specified cost function can be 15% shrunk or 15% stretched. However, the deviation from the ideal value linearly increases the price paid with the change. The cost function in the example is used by the speech descriptor (either with American or British accent) to inform how much the speech audio may be adjusted. When combined with the document temporal constraints, cost functions give metrics for the formatter finding the optimum temporal configuration. HyperProp uses tension graph formalism and a solver utility (*solve tension*) to perform the optimized computation [1].

Although not shown in the example, document link relationships establish that the explanation node switch should begin in parallel with the video presentation, and they should finish together. Since the video should last for 30 seconds (line 24), the selected alternative in the node switch must be presented during the same time. If the text is selected this can be easily accomplished. On the other hand, if the synthesis is performed, the ETHs adaptation capability (Sect. 2.2) is used to accommodate the speech duration.

Once the document execution starts, the formatter stays observing the dynamic context information and monitoring the presentation event occurring instants (time instant that transitions in the event state machines occur), and compares them with the originally predicted values. In case of any mismatch, on-the-fly elastic time adjustments are called.

To minimize the chances of needing runtime adjustments, the HyperProp formatter builds a prefetching plan, and anticipates object preparation requests [16]. The time wasted in a presentation preparation is estimated based on statistics computed from previous navigation. Prefetching is very helpful when using the ETHs adapter since the synthesis process may require considerable time.[3] To diminish this limitation, a slightly modified ETHs adapter was developed to handle previously synthesized audios. This adapter is useful in scenarios that do not tolerate synthesis delays, or when the formatter cannot have access to the synthesizer. For this new presentation tool the preparation time became about 15 times faster. However, without a speech synthesizer available, the temporal adjustment through the phonetic structure manipulation cannot be done.

Although not used in the example, another version of ETHs was developed to exclusively deal with speech synthesis. This simpler presentation tool can be used in HyperProp documents to accommodate user accessibility constraints (blind users, driver users, etc). With or without face animation, presentation tools like ETHs dynamically create alternative contents (dynamic cross-media adaptation), simplifying authoring maintenance and removing the needs for storing multiple versions.

ETHs can also be an alternative for adapting documents to devices with communication and processing limitations. Sending just the phonetic structure together with the synthesized audio requires less bandwidth than sending a face video, without excluding the visual information. Moreover, the face animation probably will require less resource usage than decompressing video algorithms.

4 Related Work

Originally proposed by Netscape, the support to plug-in technology is very common in web browsers. An API allows external applications to be integrated into the WWW navigation environment, offering a means to incorporate content formats that are not recognized by web browsers. However, the API does not offer support for defining relationships among objects presented by different plug-ins, or even between plug-ins and HTML pages. As another drawback, the plug-in lifetime is tied up to the lifetime of the HTML page containing it.

Bouvin and Schade [3] propose an extension for plug-in API aiming to allow the creation of links among objects or object internal anchors presented by different plug-ins. However, in the described extension, links are always interactive; there is no support for defining temporal synchronization relationships, like those offered by the HyperProp formatter.

Regarding media-object duration adaptation, there is some work that deals with the elastic time computation problem in multimedia presentations [9, 11]. However, none of them comments about how to apply these adjustments in continuous-media contents. The duration adjustment implemented in the ETHs adapter is a step towards this goal. The mechanism is useful to satisfy constraints specified by document authors,

[3] For instance, experiments showed a mean of 15 seconds for the synthesis of 100 words.

whenever objects with durations that do not match the inter-media specified relationships are found [1, 16]. The ETHs adjustment mechanism may also be used to compensate jitters imposed by networks and operating systems.

SMIL formatters (also known as SMIL players) [18] and the Cardio-OP application [10] are able to adapt presentations from content switches and based on contextual information. In Cardio-OP, queries [2] also allow dynamic building of document fragment alternatives. However, neither SMIL nor Cardio-OP allows duration adjustments. Actually, we are now investigating how to extend SMIL to accommodate elastic time computation algorithms and how to make use of Expressive Talking Heads in SMIL presentations.

It is possible to find in the literature researches dealing with facial animation issues, lip-sync, facial emotions, etc. VideoRewrite [4] and MikeTalk [6] are two examples that produce a very good result in lip movements. However, they are not interactive, that is, they require a large amount of time in order to prepare the animation and generate a pre-compiled video containing the result, what makes very difficult the data edition for subsequent adjustments. Reference [7] describes an implementation of a tool for supporting talking head applications in the web, using MPEG-4 as the basis for animating the faces. Although offering the support, the authors lead the implementation of user applications to a future work.

5 Conclusions

This paper describes the integration of a talking head tool into an adaptive hypermedia formatter. The talking head adapter developed in the context of this work offers a simple solution for the complex task of adjusting continuous inter-related media object durations. As an example, in the MPEG system standard it is difficult to stretch or shrink an audio stream maintaining it synchronized with a video stream. In Expressive Talking Heads (ETHs), face movements (animation) are synchronized with synthesized speech. Integrated into the HyperProp formatter, ETHs content presentation can also be synchronized with other media objects (slides, figures, etc.). Since in ETHs the visual output is synthesized based on the audio information, and the later has a well-known phonetic structure, adjustments in the second structure naturally reflect on the other one.

The presence of a talking head tool cooperating with hypermedia presentation systems increases the expressiveness of document presentations and scene descriptions. We are now working on the migration of the HyperProp formatter to ITV set-top box platforms. Our first adaptation target is non-profit advertisements (for example, programming advertisements of the broadcaster itself), in a sequence of commercials, which can be elastic-time adjusted in order to satisfy the programming grade time constraints. A more careful analysis of the overhead imposed by the synthesis process is also left as a future work, besides investigating algorithms and heuristics that better accommodate prefetching and on-the-fly adaptation requirements.

References

1. Bachelet B., Mahey P., Rodrigues R.F., Soares L.F.G.: Elastic Time Computation for Hypermedia Documents. VI Brazilian Symposium on Multimedia and Hypermedia Systems - SBMídia'2000, Natal, Brazil, (2000) 47-62
2. Boll S., Klas W., Wandel J.: A Cross-Media Adaptation Strategy for Multimedia Presentation. ACM Multimedia, Orlando, USA, (1999)
3. Bouvin N.O., Schade R.: Integrating Temporal Media and Open Hypermedia on the World Wide Web. Eighth Int. World Wide Web Conference, (1999) 375-387
4. Bregler C., Covell M., Slaney M.: Video Rewrite: Driving visual speech with audio. SIGGRAPH'97, Los Angeles, USA, (1997)
5. Dutoit T. et al.: A Short Introduction to Text-to-Speech Synthesis. Technical Report, TTS Research Team, TCTS Lab, Faculté Polytechnique de Mons, Belgium, (1997)
6. Ezzat T., Poggio T.: Visual Speech Synthesis by Morphing Visemes. Technical Report 1658, Center for Biological & Computational Learning and the Artificial Intelligence Lab, Massachusetts Institute of Technology, USA, (1999)
7. Gachery S. Magnenat-Thalmann N.: Designing MPEG-4 Facial Animation Tables for Web Applications. Multimedia Modeling Conference, Amsterdam, Netherlands, (2001)
8. Gamma E. et al.: Design Patterns: Elements of Reusable Object-Oriented Software. Addison Wesley, (1995)
9. Kim M., Song J.: Multimedia Documents with Elastic Time. ACM International Conference on Multimedia, San Francisco, USA, (1995)
10. Klas W., Greiner C., Friedl R.: Cardio-OP – Gallery of Cardiac Surgery. IEEE International Conference on Multimedia Computing and Systems, Florence, Italy, 1999.
11. Layaïda N., Sabry-Ismail L., Roisin, C.: Dealing with uncertain durations in synchronized multimedia presentations. MM Tools and Applications Journal, 18(3), (2002) 213-231
12. Lucena P.S. "Expressive Talking Heads: uma Ferramenta com Fala e Expressão Facial Sincronizadas para o Desenvolvimento de Aplicações Interativas". *Master Dissertation*, Dep. Informática, PUC-Rio, Rio de Janeiro, Brazil, (2002). (*in portuguese*)
13. Muchaluat-Saade D.C, Silva H.V.O., Rodrigues R.F., Soares L.F.G.. "NCL 2.0: Exploiting and Integrating New Concepts to Hypermedia Declarative Languages". Technical Report, Lab. TeleMídia, PUC-Rio, Rio de Janeiro, Brazil, (2004). Available in ftp://ftp.telemidia.puc-rio.br/pub/docs/techreports/2004_02_muchaluat.pdf
14. Perlin K.: Responsive Face, Media Research Lab, New York University, USA, (1997), Available in http://mrl.nyu.edu/~perlin/ demox/Face.html.
15. Rodrigues R.F., Rodrigues L.M., Soares L.F.G.: A Framework for Event-Driven Hypermedia Presentation Systems, Multimedia Modeling Conference - MMM'2001, Amsterdam, Netherlands, (2001) 169-185.
16. Rodrigues R.F., Soares L.F.G.: Inter and Intra Media-Object QoS Provisioning in Adaptive Formatters, ACM Symposium on Document Engineering, Grenoble, (2003)
17. Watt A., Taylor P., Caley R.: The Festival Speech Synthesis System: System Documentation. Technical Report, University of Edinburgh, Scotland, (1999).
18. World-Wide Web Consortium.: Synchronized Multimedia Integration Language (SMIL 2.0) Specification. W3C Recommendation, (2001).