Multiple Exhibition Devices in DTV Systems

Luiz Fernando G. Soares Romualdo M. R. Costa Marcio F. Moreno Marcelo F. Moreno
Pontifical Catholic University of Rio de Janeiro
Rua Marquês de São Vicente 225
22453-900 Rio de Janeiro, RJ, Brazil
+55 21 3527-1500 Ext: 3503

{lfgs, romualdo, mfmoreno, moreno}@inf.puc-rio.br

Multiple Exhibition Devices in DTV Systems

Luiz Fernando G. Soares Romualdo M. R. Costa Marcio F. Moreno Marcelo F. Moreno
Pontifical Catholic University of Rio de Janeiro
Rua Marquês de São Vicente 225
22453-900 Rio de Janeiro, RJ, Brazil
+55 21 3527-1500 Ext: 3503
{lfgs, romualdo, mfmoreno, moreno}@inf.puc-rio.br

ABSTRACT

Nested Context Language (NCL) is the declarative language of the Brazilian Terrestrial Digital TV System. NCL is part of ISDB (International Standard for Digital Broadcasting) standards and also the ITU-T Recommendation H.761 for IPTV services. This paper presents, discusses, and illustrates the NCL hierarchical control model for multiple exhibition device support. Based on this model, multiple devices are orchestrated to run a DTV application in cooperation. Two types of multiple device exhibitions are distinguished. Those where the same content is shown in a set of devices under a unique control, and those where content is under each individual device control, working completely independent. In this last case, depending on viewer interactions, the resulting presented content can differ from a device to another. Examples of NCL applications using both options are presented and discussed.

Categories and Subject Descriptors

I.7.2 [**Document and Text Processing**]: Document Preparation - *languages and systems, markup languages, multi/mixed media, hypertext/hypermedia, standards.*

D.3.2 [**Programming Languages**]: Language Classifications - *specialized application languages.*

General Terms

Design, Languages, Standardization.

Keywords

NCL, Ginga, digital TV, multiple devices, synchronization, DTV middleware.

1. INTRODUCTION

One of the main objections against interactive applications comes from content producers that are not willing to see their masterpieces polluted with additional pop-up information. In addition, viewer interactions usually annoy other viewers seeing the same program at the same place. The solution to this problem can come from using multiple devices working in cooperation in the exhibition of a DTV application.

Just an example, let it be a broadcasted film, indeed an animation, about a famous soccer player. During the film, a soccer shoes advertisement may be shown as result of a viewer interaction. Figure 1 shows this DTV application in a TV set, where, upon a viewer interaction, the window showing the film is resized (upper left corner) to give space to another video, an advertisement, and to a form, used to buy the product.

Figure 1 – Single exhibition device.

Let us now assume a scenario where each viewer has its own interactive device and that each device has a small screen. Using multiple exhibition devices in cooperation, the same application can run without annoying viewers that do not want to see the advertisement, and without pollute the broadcasted content, as shown in Figure 2.

Moreover, in this new scenario, different viewers would be able to navigate in their private device, doing purchases in parallel. This would be impossible using just one screen.

Figure 2 – Multiple exhibition devices.

Support to multiple exhibition devices is also important for applications that demand collaborative interactive actions among viewers, like is usual in educational games. Let it be, for example, a TV program that allows a group of users to play against another group. During the game, the result of each play may be shown in the TV set, but each group can also be allowed to see different private screens in its own interactive devices, allowing the group to exchange information without the other group knowledge.

Social TV applications, recommender systems in particular, also demand multiple exhibition devices to cooperatively run an application. Through their personal devices, viewers can generate recommendation messages, which may be sent to members of their social network [8][9].

The use of multiple devices by DTV applications is not a novelty, mainly if we talk about the use of multiple input devices [8]. Even the use of multiple independent exhibition devices is not new, mainly in Social TV applications [8]. This paper, however, is not interested just in the use of multiple exhibition devices that can communicate among themselves, but how these devices can be orchestrated in a cooperative (or distributed) exhibition of a DTV application. Thus, cooperative exhibition is the keyword when searching for the multiple exhibition device solution.

From the aforementioned examples, two types of multiple device exhibitions can be distinguished. Those where the same content is shown in a set of devices under a unique control (centralized or distributed, but unique), as for example when a game is being played among viewers and all of them see the same visual result coming from a play action; and those where content is under each individual device control, working completely independent, as in the purchase example of Figure 2. In this last case, depending on viewer interactions, the resulting presented content can differ from a device to another.

A class of devices that must present the same content under a unique control is called in this paper a class of passive type, or simply a *passive class*. A class of devices that runs the same initial content but with individual and independent control by each of its members is called in this paper a class of active type, or simply an *active class*.

As there will be applications in which part of its content is to be shown in a set of devices, another part in another set, and so on, as well as there will be applications requiring support of active and passive classes of devices, a device may be required to pertain to more than one class and to more than one class type.

As a consequence, an application author must be able to specify which part of an application should go to each class of devices. To work with class of devices during the specification phase is crucial. This allows authors to be unaware of the number of devices in a given class, which can vary over time.

On the other hand, the presentation environment should offer a procedure to register devices in classes. In a home network, for example, a media center (or a set-top box) can be responsible for this task. They must also provide some way of communication in the device domain.

When using multiple exhibition devices input units sometimes refer to the content exhibit in a screen sometimes in another one. In Figure 2, for example, before the advertisement presentation period, the interaction coming from the secondary device (in this case an iPhone) is used to navigate in the content presented in the TV set, controlled by the set-top box shown in the figure (the iPhone acts as a TV remote control). In other words, inputs coming from the secondary device are passed to the set-top box. However, when the soccer shoes advertisement is presented, interactions coming from the secondary device is not resolved by the set-top box anymore, but by procedures running on the secondary device itself.

As a general consequence, there must be an interaction model to be followed when running an application that must be presented in multiple devices acting in cooperation.

This paper discusses all these aforementioned issues, presenting the solution supported by Nested Context Language (NCL) [22] and Ginga-NCL [1]. NCL is the declarative language of the Brazilian Terrestrial Digital TV System (SBTVD) supported by its middleware called Ginga [1]. NCL and Ginga are part of ISDB (International Standard for Digital Broadcasting) standards (the previously known Japanese standard now increased with Brazilian improvements) [1]. NCL and Ginga-NCL are also an ITU-T Recommendation for IPTV services [18]. NCL and Ginga-NCL were designed at the TeleMidia Lab at PUC-Rio. The work has been coordinated by the authors of this paper that also chaired the ITU-T H.761 and the Brazilian DTV Middleware Working Group.

NCL is defined as a glue language to relate media objects in time and space. NCL does not restrict or prescribe its object content types. In this sense, image objects (GIF, JPEG, etc.), video objects (MPEG, MOV, etc.), audio objects (MP3, WMA, etc.), text objects (TXT, PDF, etc.), imperative objects (Xlet, Lua, etc.), declarative objects (XHTML, SMIL, SVG, etc.), etc., are supported by the language. Which objects are supported depends only on which object players (engines) are coupled to the NCL formatter (player)[1].

The remainder of this paper is structured as follows. Section 2 briefly discusses some related work. The hierarchical control model used by Ginga-NCL is presented in Section 3; in this section the declarative support offered by NCL is also presented. Section 4 discusses the behavior of devices registered both in passive and active classes, taking back the example of Figure 2. Section 5 is reserved to final remarks.

[1] NCL renderer, NCL user agent, and NCL player are other names used with the same meaning of NCL formatter.

2. RELATED WORK

SMIL (Synchronized Multimedia Language) [25] and MPEG-4 [17] are relevant technologies related with synchronized multimedia applications. SMIL is a W3C standard that enables authoring multimedia documents in the Web. MPEG-4 is a superset of standard technologies focused on multimedia representation, distribution and exhibition.

SMIL 3.0 defines the MultiWindowLayout module, which contains elements and attributes to describe multiple sets of regions, where multimedia content is presented. This approach is similar to the NCL solution. However, SMIL does not allow authors to specify the association between a defined set of regions to a specific device or class of devices. The association can be done using a metalanguage interpreted by some other engine, or without author interventions (automatically), through using an algorithmic procedure (based on environment features, viewer preferences and presentation context [9]). This last case usually requires complex algorithms and it is usually viable only for simple scenarios.

By using no more than SMIL constructs the authoring of applications to be exhibited over a set of devices working cooperatively is not favored. However, some workaround [6][7][9] has been proposed to add support for multiple devices in DTV systems using the facilities provided by the Ambulant open-source player [5] and the Ambulant Annotator [6]. The first one aims to be a reference implementation of the SMIL 3.0 presentation engine, while the second one is an Ambulant extension to allow viewer side enrichment of multimedia content. These two components are part of the server side in a client-server architecture [6][7], in which exhibition devices compound the client sides.

In the aforementioned client-server architecture, a personal digital recorder (PDR) saves free-to-air audiovisual broadcasted content and, eventually, content received on demand by accessing an interactive channel. All audiovisual recorded content may be automatically segmented, in a process guided by metadata referring to the saved content [6]. Although the paper's authors have not detailed the whole process, based on examples shown [6][7], it is possible to conclude that the resulting segmented content is wrapped into a SMIL application that may be presented in a set of devices attached to the server.

SMIL applications built by the server allow intra-program content navigation [6], that is, content navigation and selection from a specific segment of the saved content. This feature is supported by a set of interfaces in client devices, which are mapped to navigation and selection actions over the SMIL presentation. Actions performed at clients are transmitted to the server in order to be processed by the Ambulant player, in a real centralized architecture. Using a client device interface, it is also possible to enrich the presentation. Viewers can make content annotations and they are also allowed to add personal content to the SMIL application built by the server. Once more, annotations performed at client sides are transmitted to the server where they are processed by the Ambulant Annotator.

The asynchronous sharing and enrichment proposed in the aforementioned architecture, allows viewers to interact with media over broad time interval [9]. However, the proposed multi-device support cannot be extended to a synchronous sharing and enrichment.

The architecture modus operandi has some similarity with the active class behavior proposed in NCL, in which the same initial content is presented in all devices, but individual and independent control are allowed to each one. However, in the proposed architecture, the SMIL application control is centralized by the Ambulant player running in the server side. In Ginga-NCL the same approach could be adopted, but support is also provided to distributed control, if client devices are able to run NCL formatters. As a consequence, in NCL a client may also become a server, defining a hierarchical peer-to-peer model, as discussed and exemplified in Sections 3 and 4 of this paper. Moreover, a unique application can have parts of it spread over multiple devices that cooperate in a joint exhibition of the application (NCL document presentation).

In addition to devices in active classes (centralized or distributed controlled), NCL also allows to register devices in passive classes. During an application execution, the NCL formatter is responsible to find which devices are registered in which classes. All devices in a same class will then receive the same content to be presented following the author's specification. Thus, without needing an extra metalanguage interpreter, NCL formatter provides an author support for content distribution. Of course, the use of a metalanguage for content distribution, as well as the use of an algorithm for content distribution can also be used, together with the solution presented in this paper.

Although NCL hierarchical control model defines only two class types, the number of classes is not limited. Therefore, different content (part of an application) can be shared by different group of viewers, in other words, the shared content is not limited to the whole audience.

NCL hierarchical control model can also be applied to synchronous content sharing and enrichment. Moreover it is not limited to enrich video content; it can also enrich DTV application content, that is, in the SBTVD case, enrich broadcasted NCL applications, in which the main video of a TV program is just one of the NCL's content objects. In order to support enrichments, NCL specifies a set of editing commands [10] allowing to insert, to delete or to alter NCL entities on-the-fly. In particular, NCL live editing performed on the viewer-side allows the fragmentation[2] of the audiovisual content received on-the-fly.

The MPEG-4 System [19] allows application specification using a declarative language called XMT-O [19]. Another MPEG-4 System format called BIFS (Binary Format for Scenes) [19] is used to control the presentation, from servers to receivers. BIFS presentations are organized in scenes, each one structured as an object hierarchy. MPEG-4 LASeR (Lightweight Application Scene Representation, part 20) [20] defines a format most suitable to mobile devices, based on SVG (Scalable Vectores Graphics) [24] and other extensions [20]. The LASeR declarative specification follows the SVG scene structure that can be fragmented in many access units, each one describing the time

[2] In NCL a media object fragment is represented by a <area> element that can represent a spatiotemporal interval.

scene elements are needed by the LASeR player. When applications (declarative specification and referenced media) are fragmented in access units, they can be streamed to and presented by players.

Like BIFS, LASeR emphasizes the composition of media objects on one rendering device. Thus, they do not have a specific notation for multiple exhibition devices. However, considering that a LASeR declarative specification can be fragmented, this standard could be extended to support presentations in a distributed environment. In [12] the authors propose a time-fragmentation of SVG documents to control the playback memory usage. Indeed, some proposed ideas could also be used to develop some experiences over LASeR in multi-devices, including a data structure to help on the fragmentation task. Fragmentation strategies of LASeR applications could be used to guide their distribution over multiple devices.

Some digital TV open standards provide Java-based and XHTML-based languages to specify the synchronism and the interactivity present in their applications. XHTML-based languages, such as those used in BML [2], ACAP-X [3] and DVB-HTML [14] do not have support to multiple devices. Java-based languages can offer classes and interfaces to support multiple devices environments, exemplified by the enhanced API defined in Ginga-J [23], the imperative environment of the Ginga middleware.

Ginga-J allows input interactions and output rendering using multiple devices. To accomplish this task, several Java classes and methods are offered, allowing registering devices in a domain, listening input actions triggered from these devices, dispatching selected content to specific devices, etc. Using this API, imperative applications can be developed using a support similar to the one offered by the NCL approach presented in this paper. However, the offered Java API has a very low abstraction level. Thus, when developing the imperative application, authors need to implement the whole control model (for each application), which must be transmitted as part of the application. It will be necessary authors not only with Java expertise but also with system control expertise.

As opposed, the declarative solution presented in this paper offer a high level abstraction that relieves authors in designing applications not only focused on of the usual tasks present in DTV applications but also focused on using multiple devices. Actually, NCL applications may be used in any environment with multiple devices running in cooperation. As an example, applications following the framework proposed in [21] or with the features proposed in [13] could be specified using NCL, with the advantages of the declarative approach.

Besides the language support for multiple device orchestration, several other proposals deal with other important related issues. In [16] the authors present an environment called TERESA, which deals with multimodal authoring and multimodal presentation. In TERESA, applications are specified using an abstract notation that is translated to concrete specifications, each one guiding the presentation on a specific platform in a multiple device environment. However, when many different platforms and languages are supported, the expressiveness of the abstract notation can be lost in the translation process to poorer language models.

Much work has been done in using metadata in interactive TV [15][4]. NCL metainformation module [1] allows defining metadata following different standards. Content adaptation and adaptation of content presentation can result from actions based on these metadata. In particular, an application can be adapted depending on if it is being presented in a multiple device environment or not, depending on the class type of the devices, etc. The use of metadata in the context of multiple devices is a topic to be explored in NCL multiple device applications.

3. MULTIPLE EXHIBITION DEVICES IN NCL

An NCL application may have its exhibition spread over multiple exhibition devices working in cooperation. A hierarchical control model establishes the basis of the modus operandi.

Any device able to process exhibition tasks is called a *processing device*. Processing devices may have output and input units. A processing device together with its output units is called an *exhibition device*. Note that nothing is still said about input units, left for Section 3.2. Also note that although an exhibition device is a processing device, the reciprocal is not necessarily true.

An NCL application specifies all its processing devices joining them into classes. The processing device that runs an NCL formatter responsible for processing the application's NCL document is called the *base device*. All processing devices that jointly run an NCL application compound what is called the application's *device domain*.

Since NCL applications can embed NCL media objects, that is, media objects whose content are other NCL applications, each NCL media object has its own base device and its own domain. This domain is indeed a subset of its parent NCL application's domain; and the NCL media object base device is not necessarily the base device of the parent NCL application.

3.1 Types of Classes and Class Registration

There is no limit for the number of device classes in an NCL application. However, there are only two types of device classes: passive and active.

Active classes are those whose members are able to run media players (including players for NCL objects with imperative and declarative code span as media content).

Passive classes are those whose members are not required to run media players. However, the members shall be able to present the raw video frame buffer and the audio sample sequence they receive from other processing devices, called *parent devices*.

In NCL, the <regionBase> element can be associated with a device class where the presentation will take place. The <regionBase> element defines exhibition regions in the exhibition devices registered in the class. In order to identify the class (i) of devices, the *device* attribute of the <regionBase> element can receive the "systemScreen(i)" or the "systemAudio(i)" value. When this attribute is not specified, the presentation must take place in the same exhibition device in which the NCL document is running.

Processing devices can register themselves in classes ("systemScreen(i)" and "systemAudio(i)") of a given domain. The number of classes of a given domain can be obtained from the

settings node's[3] "system.classNumber" property. A processing device may register itself in more than one class.

A class of active type shall define which media object[4] players are available in all its registered devices.

In SBTVD, the Ginga middleware defines some default classes, for simplicity. The "systemScreen(1)" and the "systemAudio(1)" classes are reserved as classes of passive type. The "systemScreen(2)" and "systemAudio(2)" classes are reserved as classes of active type, in which all registered devices must be able to exhibit any media object type specified in the SBTVD Standard, including imperative NCLua and NCLet players[5] and declarative NCL and HTML players. Moreover, when there is just one exhibition device in the device domain, no registration is necessary. Indeed, a base device has its own class, where no other device can be registered and which is declared by default.

Ginga reference implementation [1] allows creating classes and registering in classes of the device domain where its NCL formatter runs. A Ginga's resident application is responsible for performing this task.

In NCL, the raw video frame buffer (or the audio sample sequence) created by a parent device to be exhibited in a passive class may also be exhibited in a region of the parent device. This region is specified in the *region* attribute of the <regionBase> element that defines the passive class. This attribute shall refer to a region defined for the parent exhibition device.

In the NCL hierarchical exhibition control model, independent from the class type, exhibition devices in one class can only exhibit media objects coming from the same parent device (explicitly or rendered in video frame buffers or audio sample sequences). A class may not have more than one parent device in a given moment. Moreover, a base device may not receive objects (explicitly or rendered in video frame buffers or audio sample sequences) from other devices of the same device domain. In other words, it is not possible to have a device as an ascendant or a descendent of itself.

A time snapshot of the hierarchical exhibition control model defines a tree structure processing distribution. Parent nodes can not receive objects from their children in order to avoid several undesirable behaviors, for example, loop occurrences. Content presentation superposition is also avoided, since following the tree structure a device would not have more than one parent device in a given moment. In addition, the tree structure also defines to which device an interaction input must be addressed preventing confusing navigation procedure (especially remote control key navigation), as presented in the next section.

3.2 Control of Input Units

In the beginning of an NCL document presentation, all input units associated with devices of the document device domain are under control of the domain's base device.

Turning back to the example of Figure 2, in the beginning all input units (the iPhone and the TV remote control) are under the set-top box control.

When a <media> element in exhibition on a device of an active class receives the presentation focus and is selected, the device in charge of the exhibition gains control of all its input units and all input units of devices that are in classes that will be its descendents (classes for which it will be the parent device). The media player can then follow its own navigational rules. In Ginga, when the "BACK" key is pressed, the control of all previously mentioned input units is returned to the parent device. As usual in NCL, the focus will go to the element identified by the *service.currentFocus* attribute of the settings node (<media type="application/x-ncl-settings).

In Figure 2, assuming that the advertisement is presented in devices of an active class, once the advertisement is presented, the control of the input unit associated with the iPhone is passed to the application running on the iPhone. The set-top box continues with the TV remote control. The control of the input unit associated with the iPhone is turned back to the set-top box when the "BACK" key is pressed in the iPhone virtual keyboard.

It must be noted that there can be more than one device controlling the key navigation, but each one controlling different input units from the others.

4. BEHAVIOR OF EXHIBITION DEVICES

In order to exemplify the behavior of exhibition devices when playing an NCL application, the scenario of Figure 2 is detailed as follows:

1. A video, an animation, about a famous soccer player exhibited in a primary device (a TV set, in Figure 2)
2. During the video an advertisement about a soccer shoes will be played in a class of secondary devices.
   a. When the advertisement is ready to be presented an icon will appear in the TV set (right upper corner in Figure 2), during a certain period of time, in order to notify the existence of secondary exhibition.
   b. At the same period of time a soccer shoes icon will appear on secondary devices. If a viewer selects this icon, an advertisement video and an HTML form will appear on secondary devices over a background picture, and the icon presentation in these devices will stop.
   c. The end of the advertisement video will end the exhibition on secondary devices.

Figure 3 presents a structural view of this scenario showing the application's media objects and their temporal relationships.

---

[3] NCL settings node (<media type="application/x-ncl-settings"...> element) defines a group of global variables whose scope can be all applications in a device domain, all applications defined by a TV channel, all applications defined by a TV service, etc. Depending on the scope, an application can access and change values of these variables.

[4] In NCL, an application is defined relating media objects in time and space. Media objects define content, and a set of properties for the media content presentation.

[5] NCLua objects are NCL <media> elements that contain Lua (http://www.lua.org) code. Lua is the scripting language of NCL. NCLet objects are NCL <media> elements that contain Java (http://java.sun.com) code.

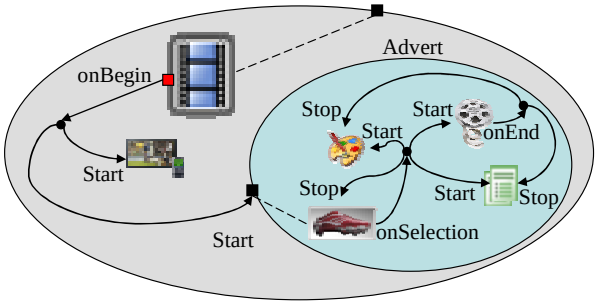

Figure 3 – Structural view of an NCL application.

4.1 Device Behavior in a Passive Class

In NCL, <regionBase> elements specify the initial placement of every object in an application. As an example, Figure 4 shows the NCL specification for the initial positioning of every media object included in Figure 3. Initial positioning are defined by <region> elements. A media object (represented in NCL by a <media> element) is associated with a presentation region through a <descriptor> element. Section 4.4 presents an important role played by <descriptor> elements when multiple devices are used.

```
<regionBase>
  <region id="screenReg" width="100%"
                      height="100%" zIndex="1">
    <region id="secIconReg" left="87%" top="11.7%"
        width="8.45%" height="6.7%" zIndex="2"/>
  </region>
</regionBase>
<regionBase device="systemScreen(1)">
  <region id="backgroundReg" width="100%"
                      height="100%" zIndex="1">
    <region id="iconReg" width="100%"
                      height="100%" zIndex="2"/>
    <region id="shoesReg" left="5%" top="30%"
        width="40%" height="40%" zIndex="2"/>
    <region id="formReg" left="50%" top="5%"
        width="45%" height="90%" zIndex="2"/>
  </region>
</regionBase>
```

Figure 4 – Passive "systemScreen(1)" class specification.

In Figure 4 two region bases, and thus two classes of devices, are defined. The first one defines presentation regions for the primary device (which is the base device declared by default). In this device, the video occupies the whole TV screen and the icon is positioned according to its left, top, width and height attributes. The second class defines presentation regions for secondary devices. The background presentation region and the icon presentation region occupy the whole secondary screen. The soccer shoes advert and the HTML form are positioned according to their left, top, width and height attributes. As aforementioned, in Ginga-NCL, "systemScreen(1)" is defined as a passive class, by default.

As mentioned in Section 3.1, the parent device is in charge of running media players for media objects to be exhibited in passive classes it controls. Based on outputs of these players, the parent device must create and transmit video frame buffers (memory maps of the exhibition plans), in the case of visual output, and audio sample sequences, in the case of audio output to its child devices (in passive classes). On the other hand, a device in the passive class must de able to scan the received video frame buffer and the received mixed audio samples, and present them. In the example, the video frame buffer is built in a set-top box connected to the TV set. The scanners for this frame buffer processes in iPhones devices.

It is important to note that when an NCL's media object has to be exhibited on devices included in a passive class, only one exhibition instance of the object must be created. This instance, created by the parent device (and included in the video frame buffer or the audio sample sequence), is then shared by all child devices in the passive class.

It should be mentioned that there is no zIndex[6] associated with the created video frame buffers or audio sample sequences. They will be presented with zIndex = 0, that is, the lowest zIndex value. The zIndex values defined in regions associated with passive classes are only used to create the video frame buffer. If a device receives more than one video frame buffer only the last one will be presented. Analogous procedure works for mixed audio sample sequences.

If a <media> element in exhibition in a passive class receives the navigational focus, its media player in the parent device gains control of all input units controlled by the parent device. From then on, this media player can follow its own navigational rules until the control is returned to the parent device by pressing the "BACK" key.

In the example of Figure 3, if the soccer shoes icon is selected by any secondary device input unit, the soccer shoes video, the HTML form and the background picture will appear in every device registered in "systemScreen(1)". If the HTML form is selected in any of these devices, any navigation inside the form, made by any input unit associated with any exhibition device, will be displayed in every device registered in "systemScreen(1)". If an individual navigation on the form is desired, the solution presented in the next section must be assumed.

The whole NCL specification for this example can be obtained from http://club.ncl.org.br. The Ginga reference implementation for the set-top box can be obtained from http://www.softwarepublico.gov.br and the passive class implementation for iPhone from http://www.telemidia.puc-rio.br/~iphone.

Figure 4 shows how easily presentation on multiple devices can be defined in NCL. Only one element (the <regionBase device="systemScreen(1)">) had to be added to distinguish the multiple device presentation from the one presenting all media objects in the TV set.

In NCL, the video frame buffer presented in secondary devices can also be shown in a region of the parent device. For that, it is sufficient to add an attribute to the <regionBase> element that defines the "systemScreen(1)" class, referring to a region defined in the <regionBase> element associated with the primary device, as shown in bold face in Figure 5.

[6] zIndex is an NCL attribute that specifies the superposition priority. The presentation of an NCL object with higher zIndex superposes the presentation of an object with lower zIndex.

```
<regionBase>
  <region id="screenReg" width="100%"
                         height="100%" zIndex="1">
    <region  id="memoryMap"  left="87%"  top="87%"
            width="10%" height="10%" zIndex="2"/>
...
  </region>
</regionBase>
<regionBase device="systemScreen(1)"
                               region="memoryMap">
...
</regionBase>
```

Figure 5 – Video frame buffer presented in the parent device.

4.2 Device Behavior in an Active Class

When dealing with an active class, the required processing for playing media object's contents is transferred from the parent device to processing devices in the active class.

As opposed to what happens in passive classes, each device in an active class must create its own media-object presentation instance for each media object it controls. In a <link>[7] element definition, <bind> elements referring to this media-object must have its cardinality max attribute set to "unbounded". Every presentation or attribution action (stop, abort, pause and resume) [22] on any of these instances must result in identical actions in all other instances (in all devices in the class) of the same media object (actions shall be performed in any order). Any condition [22] derived from these media object presentations shall be considered fulfilled, if it is fulfilled by all instances (not necessarily simultaneously), in the case its <bind> element's qualifier attribute is set to "and", or if it is fulfilled by any instance, in the case its <bind> element's qualifier attribute is set to "or".

While no <media> element[8] controlled by devices in an active class is select to gain the key navigation control, all viewer interactions through these devices' input units shall be passed to the parent device. If a <media> element is select to gain the key navigation control in a specific device X in an active class, the particular media player of this device X gains control of all input units associated with X and with all devices in descendent classes controlled by X (that have X as a parent device). From then on, the media player can follow its own navigational rules until the control is returned to the parent device by pressing the "BACK" key.

Turning back to the example of Figure 3, let an NCL media object (<media id="NCLAdvert" src=advert.ncl …>) represent all objects inside the advert context. This NCL media object (a declarative object with NCL code), defined as an NCL application apart (advert.ncl), will be presented in devices registered in an active class, as defined by Figure 6. Remember that in Ginga, "systemScreen(2)" is an active class by default.

[7] In NCL, relationships among objects are defined in <link> elements. A <link> element has <bind> children elements referring to object's interfaces that participate in the relationship, as is exemplified in Figure 7.

[8] In NCL a media object is represented by a <media> element.

```
<regionBase device="systemScreen(2)">
    <region id="NCLAdvertReg" width="100%"
                          height="100%" zIndex="1"/>
</regionBase>
```

Figure 6 – Active "systemScreen(2)" class specification.

In this example, on starting the NCLAdvert object presentation, the key navigation control is passed to it by the same <link> element used for its starting, by setting the NCL "service.currentKeyMaster" global property to this object *id*, as shown in Figure 7. This means that, from then on, each device in "systemScreen(2)" can navigate inside its NCLAdvert instance, independently and in parallel.

As aforementioned, in NCL a <link> element defines a relationship among media objects. In Figure 7, the <link id="lBeginAdvert" …> specifies that when the "animation" reaches the "segIcon" area[9] (role="onBegin"), the icon "secIcon" must start its presentation (role="start") in the TV set (see Figure 3), the NCL application NCLAdvert must also start its presentation and receive the key navigational control (role="set").

```
<link id="lBeginAdvert"
                xconnector="conEx#onBeginSetStart">
   <bind role="onBegin" component="animation"
                        interface="segIcon"/>
   <bind role="start" component="secIcon"/>
   <bind role="start" component="NCLAdvert"/>
   <bind role="set" component="globalVar"
            interface="service.currentKeyMaster">
      <bindParam name="var" value="NCLAdvert"/>
   </bind>
</link>
```

Figure 7 – Starting the NCL media object to be presented by devices in an active class.

The NCLAdvert initiates its presentation exhibiting the soccer shoes icon. Every secondary device can individually select this icon (since it now has the key navigation control). On devices where the icon is selected (and only on them), the advertisement video and form are presented over a background picture. A viewer can then purchase the soccer shoes filling the form and send it via a return channel.

The whole NCL specification for this example can also be obtained from http://club.ncl.org.br. The Ginga reference open source implementation for the set-top box can be obtained from http://www.softwarepublico.gov.br. The iPhone active class open source implementation is not yet available (is under negotiation). However the same Ginga open source implementation can be used in another Linux based device (for example a notebook) that can be registered in the "systemScreen(2)" active class.

4.2.1 Distributed Presentation

Following the NCL hierarchical exhibition control model, a media object is sent to a device in an active class from its parent device one by one, when the moment of the object presentation arrives. However there is an optional exception for this procedure.

[9] Each media area specifies a subset of the media object's content. In NCL they are defined by <area> elements, children of <media> elements they refer.

If devices in the active class are able to play NCL applications (that is, if devices implement the NCL formatter), set of objects can be received in block.

NCL Formatter uses an event-oriented hypermedia temporal graph (HTG) [11] to control an NCL application presentation. This data structure can be initially created from the application specification, and represents all events (occurrences in time) that can happen during the application execution (or presentation). Besides predictable events (like the start of a media-object presentation triggered by the end of another media-object presentation), unpredictable events (events whose occurrence in time can only be determined during runtime) are also represented, like possible viewer interactions, decision points for content and presentation adaptation, etc. From the HTG, an NCL formatter maintains all temporal and spatial relationships specified by the document author.

A temporal chain corresponds to a sequence of predictable events initiated from an unpredictable event. Temporal chains can be derived from HTG by an NCL formatter running on the parent device, which may also compute which part of a temporal chain will be exhibited in devices of an active class. If devices in this class are able to run NCL applications, the NCL formatter running in the parent device may choose to pass the entire chain, and the referred objects to be exhibited in this class, to its devices. Based on the received chain, NCL formatters are then able to instantiate objects they must control; instead of receive command to instantiate each object in the chain one by one.

4.3 Device Registered in Passive and Active Classes

As aforementioned, a device can be registered in many active and passive classes simultaneously, since all classes are controlled by the same parent device in each moment in time. In this case, the device inherits the behavior of both types of classes. In short:

- An exhibition device can only present objects passed (explicitly, or nested in other explicitly received object, or embedded in video frame buffers/audio sample sequences) to classes in which it is registered;
- A base device may not receive objects (explicitly or embedded in video frame buffers/audio samples) passed by other devices in the device domain it defines;
- When an object must be presented by devices in an active class, individual instances are created by each device in the class. Devices in the class must then behavior as specified in Section 4.2;
- When an object must be presented by devices in a passive class, a video frame buffer and an audio sample sequence are created and controlled by the parent device, which will create only one instance for each object to be presented. Child devices must behavior as specified in Section 4.1;
- There is no zIndex definition for the video frame buffer or the audio sample sequence received from the parent device. They must be presented with zIndex=0 in child devices. If more than one video frame buffer or more than one sequence of audio samples is received, only the last one must be present. Every object presentation received from an active class has its presentation placed on the top of any already received video frame buffer or sequence of audio samples.

4.4 Presentation Adaptation

As aforementioned, to work with class of devices during the specification phase allows authors to be unaware of the number of devices in a given class, which can vary over time. However this number can be zero. In this case, content forwarded to this class would not be exhibited. It would be nice if an author could specify an alternative presentation for this situation.

As mentioned in Section 4.1, a media object initial positioning is determined in NCL when a <descriptor> element is associated with a <region> element, which in its turn is associated with a class of devices. NCL allows specifying descriptors alternatives for a media object presentation [22]. An alternative is chosen during runtime based on rule evaluations. Rules are defined over global variables maintained by the NCL formatter. For each class of devices there is a global variable (system.devNumber(i)) whose value is the number of devices registered in the class, and this variable can be used to select the initial positioning of a media object.

Therefore, depending on the system.devNumber(i) value a content can be deviated to another device. As an example, if in Figure 2 the number of secondary devices was zero, the advertisement could be deviated to be presented in the TV set, superposed to the soccer match animation.

5. FINAL REMARKS

This paper focuses on how multiple devices can be orchestrated in a cooperative (or distributed) exhibition of a DTV application. The use of multiple exhibition devices in the Brazilian DTV system has just started and there is a small number of experiments. Unfortunately, the majority of commercial implementations of Ginga-NCL still do not support this feature, since the middleware support is required, but the hardware support is optional. Just one commercial implementation, as far as the authors' know, provides a real support in agreement with this paper.

The reference implementation of Ginga-NCL provides support to multiple devices in all its aspects. It is offered as an open-source GPL implementation[10]. This implementation is the basis of experiments in course in broadcasters and content producers, in particular the main Brazilian broadcaster.

We hope that in a near future these experiments can give us further idea of the potential use of multiple devices and their usability problems.

As NCL can embed media objects with imperative and declarative code, other than NCL, a future work is to have other language objects running on secondary devices, exploring the real vocation of NCL as a glue language. Together with the SMIL group of CWI, we are planning some large experiments with devices able to run SMIL applications, such as recommender systems, and others.

[10] http://www.softwarepublico.gov.br

6. ACKNOWLEDGMENTS

The authors would like to thank Roberto Gerson, Bruno Lima and Carlos Eduardo Batista who provided a thoughtful discussion of this work, tracked down and also fixed problems in the reference implementation of Ginga.

7. REFERENCES

[1] ABNT NBR 15606-2 Associação Brasileira de Normas Técnicas. 2007. Digital Terrestrial Television Standard 06: Data Codification and Transmission Specifications for Digital Broadcasting, Part 2 – GINGA-NCL: XML Application Language for Application Coding (São Paulo, SP, Brazil, November, 2007). http://www.abnt.org.br/imagens/Normalizacao_TV_Digital/ABNTNBR15606-2_2007Ing_2008.pdf.

[2] ARIB Association of Radio Industries and Business. 2004. ARIB Standard B-24 Data Coding and Transmission Specifications for Digital Broadcasting, version 4.0, 2004.

[3] ATSC Advanced Television Systems Committee. 2000. ATSC Data Broadcasting Standard - A/90, 2000.

[4] Devlin B., Wilkinson J. The MXF Boo: An Introduction to the Material eXchange Format, Focal Press, 2006. ISBN: 978-0-240-80693-8

[5] Bulterman D.C.A., Jansen J., Kleanthous K., Blom K., Benden D. 2004. AMBULANT: A Fast, Multi-Platform Open Source SMIL Player. In Proceedings of ACM International Conference on Multimedia (New York, USA, 2004).

[6] Cesar P., Bulterman D.C.A., Jansen A.J. 2006. An Architecture for End-User TV Content Enrichment. In Proceedings of European Interactive TV Conference (Athens, Greece, 2006). EuroiTV 2006.

[7] Cesar P., Bulterman D.C.A., Obrenovic Z., Ducret J., Cruz-Lara S. 2007. An Architecture for Non-Intrusive User Interfaces for Interactive Digital Television Experiences. In Proceedings of European Interactive TV Conference (Amsterdam, Netherlands, 2007). EuroiTV 2007.

[8] Cesar P., Bulterman D.C.A, Jansen A.J. 2008. Usages of the Secondary Screen in an Interactive Television Environment: Control, Enrich, Share, and Transfer Television Content. In Proceedings of European Interactive TV Conference (Salzburg, Austria, July 2008). EuroiTV 2008.

[9] Cesar P., Bulterman D.C.A., Geerts D., Jansen J., Knoche H., Seager W. 2008. Enhancing Social Sharing of Videos: Fragment, Annotate, Enrich, and Share. In Proceedings of ACM International Conference on Multimedia (Vancouver, Canada, October 2008). ACM MM 2008.

[10] Costa R.M.R., Moreno M.F., Rodrigues R.F., Soares L.F.G. 2006. Live Editing of Hypermedia Documents. In Proceedings of ACM Symposium on Document Engineering (Amsterdam, Netherlands, 2006). DocEng 2006.

[11] Costa R.M.R, Moreno M.F., Soares L.F.G. 2008. Intermedia Synchronization Management in DTV Systems. In Proceedings of ACM Symposium on Document Engineering (Sao Paulo, Brazil, 2008). DocEng 2008.

[12] Concolato C., Le Feuvre J., Moissinac J.C. 2007. Timed-Fragmentation of SVG Documents to Control the Playback Memory Usage. In Proceedings of ACM Symposium on Document Engineering (New York, USA, 2007). DocEng 2007.

[13] Dini R., Paterno F., Santoro C. 2007. An Environment to Support Multi-User Interaction and Cooperation for Improving Museum Visits through Games. In Proceedings of Mobile Human-Computer Interaction Conference (Singapore, September 2007). Mobile HCI 2007.

[14] ETSI European Telecommunication Standards Institute. 2006. ETSI TS 102 812 V1.2.2 Digital Video Broadcasting "Digital Video Broadcasting (DVB); Multimedia Home Platform (MHP) Specification 1.1.1".

[15] Lugmayr A., Niiranen S., Kalli S. Digital Interactive TV and Metadata. Future Broadcast Multimedia, Series: Signals and Communication Technology, Springer, 2004. ISBN: 978-0-387-20843-5

[16] Paterno F., Santoro C., Mantyjarvi J., Mori G., Sansone S. 2008. Authoring Pervasive Multimodal User Interfaces. In International Journal of Web Engineering and Technology, V 4, ISSN: 1476-1289.

[17] Pereira F., Ebrahimi T. The MPEG-4 Book, Prentice Hall, 2002. ISBN: 0-13-061621-4.

[18] ITU-T Recommendation H.761, 2009. Nested Context Language (NCL) and Ginga-NCL for IPTV Services. Geneva, April, 2009.

[19] ISO/IEC International Organization for Standardization 14496-1. 2004. Coding of Audio-Visual Objects – Part 1: Systems. 3nd Edition.

[20] ISO/IEC International Organization for Standardization 14496-20. 2006. Lightweight Application Scene Representation (LASeR) and Simple Aggregation Format (SAF).

[21] Schroyen J., Gabriels K., Luyten K., Teunkens D., et al. 2008. Training Social Learning Skills by Collaborative Mobile Gaming in Museums. In Proceedings of International Conference on Advances in Computer Entertainment Technology (Yokohama, Japan, December 2008). ACE 2008.

[22] Soares L.F.G., Rodrigues R.F. 2006. Nested Context Language 3.0 Part 8 – NCL Digital TV Profiles. Technical Report. Departamento de Informática da PUC-Rio, MCC 35/06. http://www.ncl.org.br/documentos/NCL3.0-DTV.pdf.

[23] Souza G. L., Leite L.E.C, Batista C.E.C.F. 2007. Ginga-J: The Procedural Middleware for the Brazilian Digital TV System. In Journal of the Brazilian Computer Society, N 4, V 13, ISSN: 0104-6500.

[24] W3C World-Wide Web Consortium. 2003. Scalable Vector Graphics – SVG 1.1 Specification, W3C Recommendation. http://www/w3/org/TR/SVG11

[25] W3C World-Wide Web Consortium. 2008. Synchronized Multimedia Integration Language – SMIL 3.0 Specification, W3C Recommendation. http://www.w3.org/TR/2008/REC-SMIL3-20081201/