3D Technologies to Extend Brazilian DTV Middleware

Daniel F. L Souza
Universidade Federal Rural do Semi-Árido
Mossoró/RN
danieltidus@gmail.com

Liliane S. Machado, Tatiana A. Tavares
Departament of Informatics
Universidade Federal da Paraíba
João Pessoa/PB
{liliane, tatiana}@di.ufpb.br

***Abstract*—Entertainment and interactivity possibilities in Digital Television (DTV) can be extended to support 3D technologies. In this paper is described an architecture based on a middleware for DTV that incorporates 3D technologies on the Brazilian standard. The integration strategies will be presented and compared with other studies in the literature. As case study, it will be presented a virtual environment to analyze the advantages and disadvantages of this integration.**

Keywords-digital television, 3D Technologies, Ginga

I. INTRODUCTION

Recently, Brazil has defined its standard of Digital Television (DTV) called ISDB-Tb. The Brazilian standard is based on the Japanese standard (ISDB-T) but adds some features that are not provided by the ISDB-T, as the use of MPEG-4 standard for video compression. In the case of the ISDB-Tb, the software layer responsible for managing and executing of interactive applications is the middleware Ginga. The Ginga offers some innovations that other middleware standards for TV in the world do not. The multiple communication devices can be cited as an example of this type of innovation.

Parallel to advances in the digitization of media and adoption of DTV standards in some countries, new models of interaction are emerging for DTV. For example, the incorporation of new technologies for building three-dimensional applications in embedded systems. These technologies are sets of standards and API's (Application Programming Interface) that aim to support the construction of three-dimensional applications. Currently, these technologies are advancing in order to combat the high computational cost demanded by 3D applications. These advances allow the use of such techniques within a DTV environment providing a new set of possibilities and ideas with regard to interactivity [1].

Virtual Reality (VR) is an area that combines computers, physics, graphics, electronics, cognition and many other concepts. A VR system is a real-time application which goal is to produce realistic applications to explore human senses [2]. The senses more explored are the vision, audition and touch, but recently have been developed technology to explore the smell and taste senses [3]. The integration of 3D technologies to DTV allows the adaptation of VR environments to this type of platform, expanding the number of possibilities for the development of applications in DTV environments. On the other hand, the use of TV for the execution of virtual environments allows access to these applications by users of TV, which represent a large part of the Brazilian populace.

This paper discusses the incorporation of 3D technologies to the Brazilian middleware Ginga by the Ginga3D. In this sense, will be presented the strategies adopted for the incorporation of 3D in TV standard as well as the advantages of using such technologies in DTV environments. This work also analyzes the development of a virtual environment based on the integration strategies in order to validate the extended architecture of middleware.

II. BENEFITS OF 3D ON DTV

The establishment of the Digital TV standard in Brazil has as one of its goals the reduction of social inequalities by providing access to information. The development of new platforms to promote the e-learning, for example, is one of the main areas of interest of Brazilian government [4]. In this sense, the integration of 3D technologies to Brazilian standard can be seen as a relevant factor, since such technologies can be used as powerful tools for the design and development of applications for social purposes. Virtual environments for collaboration activities, educational games and systems to support innovative ways of visualization are solutions that could use these technologies to offer more intuitive ways to interact with the user.

Another important feature in the Brazilian standard is the transmission and presentation of content in the LIBRAS (Brazilian Sign Language) format. The transmission of content in this language provides the accessibility of TV programs to the deaf. The presentation of such content to the viewer has not yet been defined. One of the candidates to solve this problem is the use of 3D avatars to interpret the signals sent by the broadcasters. Thus, systems for the presentation of content in LIBRAS can be supported by the incorporation of 3D systems to middleware.

ISSN: 2236-3297

Digital convergence is the term used to describe the tendency to use the same technology infrastructure in order to provide services that previously required different communication channels and patterns. This trend shows up as another motivating factor, since digital TV is one of the candidates to participate in this convergence as one of modern communication technologies. The importance that is given today to 3D technologies in various fields, as well as its use as a tool that extends the capability of individuals to interact with applications, coupled with the convergence trend that modern technologies are inserted and the need that DTV has to expand and explore their potential with regard to the means of interacting with the user, are the main motivating factors that lead us to realize the immediate need of integrating such technologies.

III. DIGITAL TV

A basic system of DTV is composed by a broadcast station, a physical medium over which the signal is transmitted, which can be air driven or physical medium driven (coaxial cable, optical fiber, etc.), and a receptor responsible for receive the broadcast signal, decode it and display it. As the transmission is made through a stream of bits, it is possible to transmit a large amount of information multiplexed in comparison to the analog system [5]. Due to this feature, the DTV systems tend to adopt standards for video coding that supports higher resolution than those available in standard analog TV, as well as standards for audio encoding for supporting a large number of channels.

A Digital transmission enables the sent of multiple formats simultaneously, so that the content can be transmitted at different resolutions for different devices. This feature is what also allows the transmission of data streams such as information systems and interactive applications. Therefore, it is necessary to establish standards that regulate the entire process of capture, compression, modulation and transmission of video signals, and all physical interfaces among the equipments involved in the process. This is done by defining a set of standards, one for each of the components that make up a digital television system [6].

Currently, there are four global standards for digital television systems. These standards differ on the diversity of technological solutions that can be adopted [6]. The global standards are: ATSC (Advanced Television System Committee) (American standard), DVB (Digital Video Broadcasting) (European standard), ISDB (Integrated Services Digital Broadcasting) (Japanese standard) and ISDB-Tb (Brazilian standard).

The Brazilian standard for DTV uses the Japanese transmission scheme. The programs are broadcasted in two formats: high definition (HDTV) for home reception and low definition (LDTV) for mobile devices. In the video coding layer is used MPEG-4 Video (H.264), the audio coding layer uses MPEG-4 AAC (Advanced Audio Coding). At the transport layer for audio, video and data is used the MPEG-4 TS (Transport Stream) [7].

An important contribution provided by the adoption of a standard for DTV, is the possibility of transmission and presentation of interactive content [8]. This interactive content is nothing else than a software that will run over the middleware embedded in STBs (set top boxes). In this context, that software or interactive applications have some special characteristics if compared to conventional applications. These differences are related to the target public of this application as well as their lifetime [9]. Other features that highlight them are the fact that they require an infrastructure of transmission and a business model, this last not yet fully defined.

A. *Middleware Ginga*

Middleware is an intermediate layer of software between the application code and operating system or hardware platform [10]. A middleware for DTV can incorporate the necessary resources to support the execution of interactive applications. Thus, it is composed by an execution machine for the programming languages, supported by the DTV standard and libraries, that enables the development using these languages.

Ginga is the standard middleware of the Brazilian Digital Television system. It is responsible for supporting the declarative and imperative applications and it's compatible with the international definitions of the ITU (International Communication Union) [11]. One feature that differentiates the Ginga from other existing middleware is the fact that it was designed to attend certain social aspects of the Brazilian society. When the Brazilian government began the research for the development of Ginga in 2004, they set some requirements based on features of the Brazilian social context. Innovative features founded at Ginga were developed for enable the creation of advanced applications that seek to explore the integration with other technologies in order to facilitate information access to the entire population. These features are included in innovation API of the middleware.

Ginga is the result of several years of research in the area of middleware for DTV. The specification was based on two surveys related to specification of environments for DTV: the MAESTRO [12] and FlexTV [13]. The first one focused on a declarative environment and the second one focused on an imperative environment. Fig. 1 shows the architecture of the middleware Ginga.

The Ginga Common Core is the layer of Ginga responsible for providing support in low-level to API's in declarative and imperative environments. It consists on a set of components built using the features of specific hardware. Some of the components that are present in this layer are the audio and video decoders, tuners and libraries for access information system of the data stream transported using MPEG-2 system.

The Ginga-NCL [14] is the declarative environment of middleware Ginga. The language in this environment is the NCL (Nested Context Language) [15], a language for

description and synchronization of media. The Ginga-NCL is composed by formatters of NCL content, which are the mechanisms responsible for decoding the NCL documents. Other components present in this layer are based on XHTML, including Cascading Style Sheets (CSS) and an interpreter for ECMAScript [16]. Finally, this layer still has a virtual machine to the scripting language Lua.

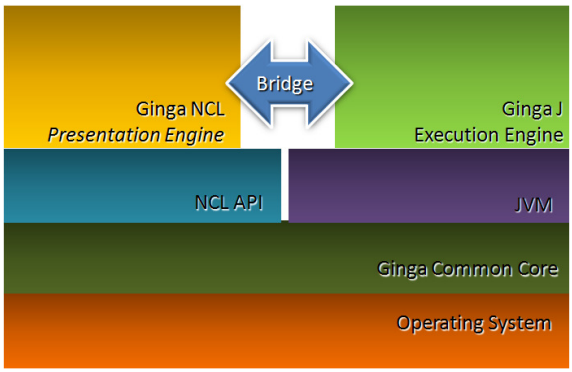

Figure 1. Architecture of the middleware Ginga.

The Ginga-J [17] is the imperative environment of middleware Ginga. The language used in this environment is Java. This environment consists on a set of Java APIs that enable the development of applications for DTV. Basically, the APIs that compose the Ginga-J are: the Service Information API of ISDB (ARIB B.23), the Java DTV specification, JMF API 2.1 and APIs to specific services, as it is possible see in Fig. 2.

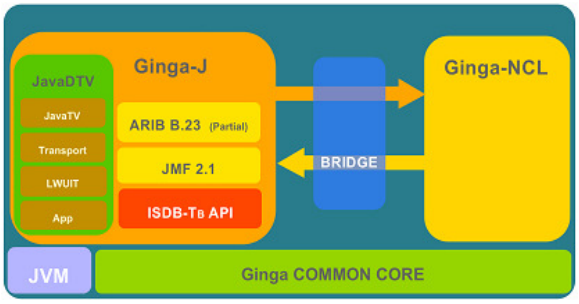

Figure 2. Ginga-J specification.

An innovation of middleware Ginga is the ability to interact with multiple devices. Other standards only provide user interaction using the TV remote control. The Ginga, however, provides the use of mobile devices to interact with applications. It allows multiple users to interact simultaneously with such applications. Thus, Ginga extends environments where only one user can interact to environments where many users can interact with the same application [18].

IV. 3D TECHNOLOGIES

The use of graphics systems gives the most natural way to communicate with computer systems, since the human capacity of recognize 2D and 3D patterns allows us to perceive and interpret image data quickly and efficiently [19]. Over the years several researches in computer graphics have been working to offer a several techniques and devices to allow interaction with 3D applications in real-time [20][21]. Parallel to the emergence of these techniques, a series of tools and APIs that enable the development of three-dimensional systems have been developed [22]. This set of techniques, tools and graphics APIs is called 3D technologies.

According to [19] two factors that, over time, contributed to the emergence and growth of 3D technologies were:

- The development of integrated circuit technology that allowed cheaper prices and the consequent popularity of the machines.
- The popularization of certain applications (spreadsheets, text editors, graphic editors, image processors, databases, etc.) resulted in the popularization of these technologies.

Advances in research related to 3D technologies have boosted the area of computer graphics through the design of tools for rapid and economic development of graphics systems. Currently, there are almost no areas where it is not possible imagine any beneficial use of such technology. Several areas such as virtual reality, data visualization, education, training, entertainment, etc. are benefited [23].

Nowadays, another area where this type of technology is being used is the Web. The Web 3D Consortium is an initiative for the development of 3D technologies that support the creation and access to 3D environments through the Internet in a practical and fast way. The use of these technologies is intended to integrate a range of services over the network, offering support and access to any user who wishes to navigate. An initiative that is growing is the X3D Earth, a project based on X3D (eXtensible 3D) technology [24] which aims to create a collaborative virtual community, enabling each user to integrate its own services in this system and collaborate to build this virtual environment.

Nowadays, there are a number of 3D APIs which depend mostly on the platform for which they were developed. These APIs were designed for the development of applications implementing 3D computer graphics techniques widely used in industry. Some examples are OpenGL and Java 3D specified for development in PC and Workstations. Other examples are the OpenGL ES and M3G developed with the aim of supporting the development of 3D applications in embedded systems.

In addition to graphics APIs already mentioned, there are other technologies that allow the specification of three-dimensional environments and enable developers to program modes to interact with these environments. The standards for describing 3D scenes are an example and use syntax to describe three-dimensional objects properties, as their position in the scene, material properties and behavior. Most of them need a browser capable to interpret the description files, graphically render the environment and perform pre-defined

events. Examples of such technologies are the VRML (Virtual Reality Modeling Language) and X3D.

V. 3D TECHNOLOGIES ON DTV

Although there are some improvements over the support of 3D applications on mobile devices, this is not a fact in the Digital TV scenario. Researches related to the support of 3D content for DTV are in an initial phase and there are few works that aim to analyze the potential of integrating these two technologies. This integration, as well as the use of Artificial Intelligence [25] and Virtual Reality, may contribute to the evolution of the concept of interactive content [26].

Some related works are already emerging in this area. Specific goals, such as the development of prospective applications, are approached in [27] [28]. Other investigation looks for a way to provide support to the transport, processing and presentation of 3D graphics [29]. The solution discussed in [26] provides a platform to intelligent interaction within an environment of DTV. In this work is discussed the use of a 3D virtual avatar to help the interaction between the user and TV. Although the focus of the work is the intelligent service, the author also discusses the advantages of using the 3D virtual avatar [26].

Other studies use the MPEG-4 standard to describe three-dimensional scenes. This is possible because the MPEG-4 standard defines a set of tools for building graphical content using a language for synchronizing media: the BIFS (Binary Format for Scene) [30]. The work of Pulles and Sasno (2004) uses the MPEG-4 as a strategy for integrating 3D graphics with TV [31]. The authors demonstrate how the MHP standard can be used, in conjunction with the MPEG-4, to provide more complex interactive content. There are other projects that focus on the use of MPEG-4 for the generation of more complex interactive content [32] [33].

The integration of 3D technologies on Digital Television through the extension of existing standards of middleware to support the development and implementation of three-dimensional applications is also the focus of researches. This strategy intends to allow user to execute 3D applications on TV. Through this integration, new possibilities of interactive applications arise in several areas. The work of Cesar (2005) [34] analyzes this strategy in terms of a new level of interactivity within the TV environment. The author also proposes an architecture platform for implementing these applications, based on standard European Digital TV, the UBIKA [35].

The UBIK was developed based on studies about extension of middleware to TV in order to provide support for execution of three-dimensional application on set top box. The author proposed a platform that included a graphics API for development of the applications, and native support for implementation of these applications. The UBIK used the OpenGL API for building applications. In the native layer SDL is used to manage the context of applications and DirectFB to display the scene due to the user. One of the goals of UBIK was to assess the requirements needed to generate 3D scenes in a TV environment. The author examined memory and processing resources required to run the environment and compared the results with existing devices.

VI. INCORPORATING 3D TECHNOLOGIES TO GINGA

As it was described in the last section, the support for 3D technologies in a Digital Television environment expands the range of possibilities for entertainment, interactivity and, as result, new business. In the context of the Brazilian DTV these technologies can assist the government and the private sector to achieve the goals set by the SBTVD (Brazilian System Digital Television) in our country. In terms of government, such technologies can support the construction of virtual environments that help in t-learning, cultural development and dissemination of information in a more attractive way. Within the private sector, the areas that could obtain significantly gains with these technologies are the gaming market and derivatives. Another factor that could be exploited by both initiatives (public and private) is the convergence of technologies, through the construction and adaptation of virtual platforms to offer multiple ways of access.

As previously presented, the Ginga is composed by a presentation engine (Ginga-NCL) and an execution engine (Ginga-J). These two modules of the middleware enable to run applications based on two different paradigms of design and application development. The Ginga-NCL uses a declarative language and the Ginga-J uses a imperative language. So, supporting 3D systems in the Brazilian middleware means finding ways to support the design and development of 3D applications based on one of these two paradigms, or even both. But the use of declarative and imperative languages has their strengths and difficulties.

The Ginga-J uses the Java programming language and its specific APIs for TV application development. To the adoption of an integration strategy that included the execution engine, the 3D API should be based on that language. In terms of development, the Java language is easy to learn and widespread in the communities of developers. Additionally, applications developed in the Ginga-J facilitate the use of business layer, useful to games and other interactive applications. Another interesting feature is related to the support of graphics APIs for the development of application. There are already libraries written in Java for embedded systems, for example, the graphics APIs for mobile devices. This is an advantage, since it is possible to adapt them to execute 3D content on TV. One negative factor related to the use of Java is the synchronization of media. At this point the language does not provide robust resources to develop applications with the focus on description and synchronization of multiple media

The Ginga-NCL uses language NCL (Nested Context Language), whose main characteristic is the mechanisms for synchronization of multiple media. This feature would allow the presentation of 3D objects synchronized to video stream

broadcasting or to other objects of media that eventually are being presented. The use of language NCL also facilitates the presentation of three-dimensional environments based on description languages, such as X3D and VRML. 3D applications built in this context would have a focus on the presentation of graphical content. In addition, a minimum of interactivity would be also possible. On the other hand, NCL has no native support for description and synchronization of 3D media. Another problem of Ginga-NCL is related to building applications with a business layer more robust, since the NCL language does not allow a simplified way to incorporate business logic to its applications.

Based on the analysis of the advantages and disadvantages of both environments, a hybrid approach was chosen. The integration and/or adaptation of 3D APIs in both environments (imperative and declarative) was proposed. This strategy provides flexibility for developers of 3D applications that can use the benefits of each environment based on the type of interactive application that they want to develop. Basically, applications that need more advanced controls for synchronization between media can be developed in the declarative environment (NCL language). Moreover, applications with focus on the business layer and requiring only basic mechanisms of synchronization can be developed in the context of imperative environmental (Java language).

A. *Ginga3D – Extending the Ginga Middleware*

The Brazilian Digital TV does not support implementation of 3D applications in its standard, preventing the development, execution and marketing of these applications. Currently, the Brazilian standard for DTV is not completed, especially the sections that discuss interactivity and supported APIs. Thus, it is possible to suggest the expansion of the standard to enable support for 3D applications by extending the architecture of middleware Ginga. The architecture of Ginga specifies that all basic services must be provided by the common core (Ginga Common Core) that makes the interface with the operating system. The layers above the common core use the services provided by it to perform their tasks. Following this methodology, the support for 3D applications in the upper layers depends on this support in the common core. After that, it is possible adopt one of the strategies for developing applications that can be based on execution engine (imperative language), presentation engine (declarative language), or both

In order to extend the current architecture of Ginga, it was proposed a specification that incorporates modules that can enable the implementation development and execution of three-dimensional applications. This extension is called Ginga3D. Its main goal is to offer the ability to execute 3D applications in a DTV environment and integrate graphics APIs and players to three-dimensional media objects, enabling the development of these applications. Thus, Ginga3D is an expanded architecture that extends the middleware Ginga in order to provide 3D support to Ginga-J and Ginga-NCL. As discussed in section 5.1, each of the paradigms used by these two modules have features that favor or limit its use and an architecture that offers support for 3D systems in both environments provides the flexibility to choice the technology to use. This flexibility is a factor that differentiates this work from related works presented. Other factor is that this strategy enables the use of the declarative environment for the specification of 3D media. This is another important contribution of the present work, since a previous one [34] use only of the imperative environment.

Basically Ginga3D acts in different layers of middleware providing the necessary abstraction for both: the user developer of applications and the user that will use the applications. In Ginga Common Core layer, for example, it is located a native module, called Native3D, responsible for rendering objects and graphic scenes (see Fig. 3). It also integrates the modules of the upper layers and provides a well-defined way for information transaction between the layers used for graphic rendering and event handling. Native3D is composed by a sub-module called ContextManager and media players (defined in another sub-module called Renderer).

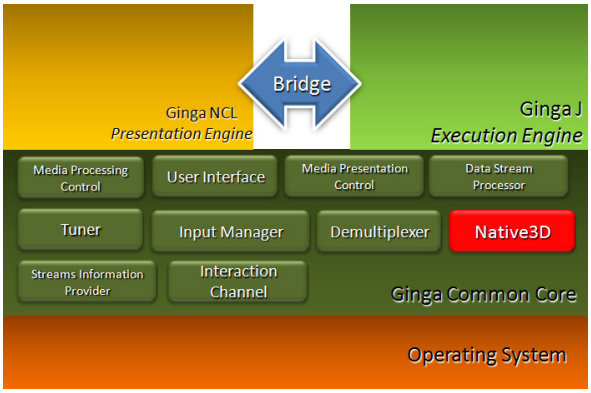

Figure 3. Native3D module in Ginga Common Core to support 3D applications.

The sub-module Renderer (see Fig. 4) encapsulates native renderers such as browsers or media players. This sub-module aims to provide an access point to the APIs in high level (NCL and Java) so they can access their native implementations. The Renderer generates three-dimensional scenes using native APIs like OpenGL, Direct3D, SDL, etc. The functionalities of these APIs are accessed through a common point that would be the sub-module ContextManager. For 3D media players necessary to NCL, the Renderer provides access to implementations of browsers that will render the virtual environment described by languages such as X3D and VRML.

The 3D player (NCL3DPlayer) is located in the intermediate layer of the presentation engine and is executed when 3D media objects are described in NCL documents (see Fig. 5). This player is compliant with the Brazilian standard for DTV that defines the use of players for the execution of media described by the language NCL [36]. This module should allow multiple types of exhibitors, so that there would be a unique access to these exhibitors through adapters. The

NCL formatter uses this player to perform calls that are defined in the standard (start, stop, resume, etc.). These calls should be consistently mapped to exhibitors allowing interpreting each of the specific commands.

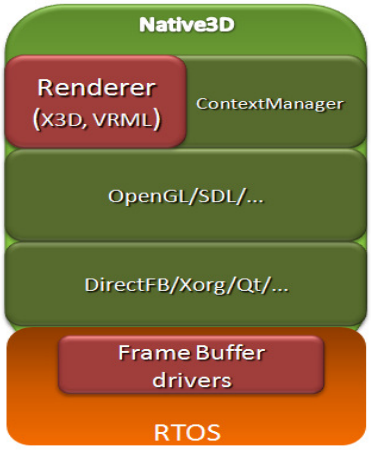

Figure 4. Sub-module renderer and others components of Native3D module.

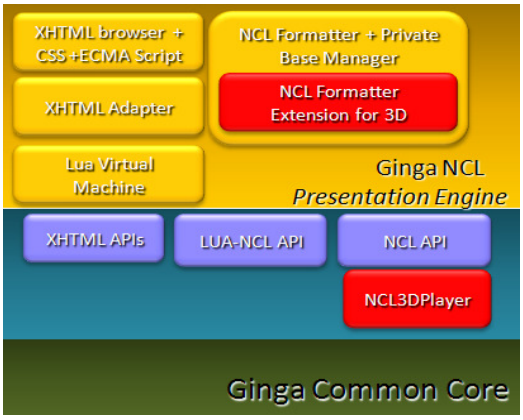

Figure 5. Incorporating a player to 3D media and extending NCL language to support 3D nodes.

A problem related to the use of 3D media within the Ginga-NCL is the fact that NCL language does not have descriptors to this type of media. So, it is impossible to include references to X3D documents in a NCL document, for example. Fortunately, this problem can be partially solved because the NCL model allows expansion of their language to include other types of media descriptors. Through the specification of new descriptors for 3D media and determination of the temporal relationship it is possible to write NCL documents that include 3D nodes.

At the Ginga-J it was proposed the use of some widely used 3D API with a focus on embedded systems such as the OpenGL ES or M3G. The OpenGL ES graphics API was chosen to integrate the Ginga-J, since it is a robust standard designed for use in embedded systems. The use of OpenGL ES also allow to incorporate graphics cards that implement the OpenGL ES standard within set top boxes. The Ginga-J incorporates to its set of libraries the API JSR 239, the java bind to OpenGL ES. However, changes were not necessary in the OpenGL ES specification, since it has mechanisms to deal with various window systems through EGL (Embedded-System Graphics Library) interface. In terms of event handling, there are already others libraries in Ginga-J to this purpose.

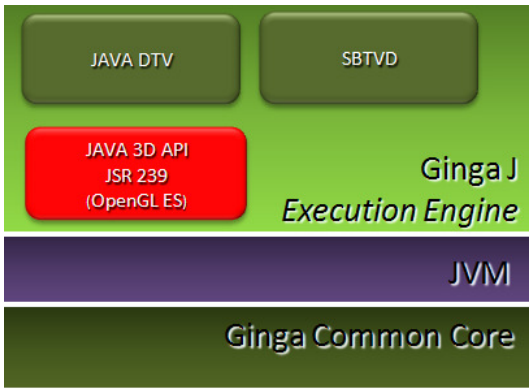

Figure 6. Inclusion of JSR 239 (Java Bind for OpenGL ES) in Ginga-J.

Based on the architecture of the Ginga, the JSR-239 was incorporated into the set of libraries that are part of the Ginga-J. It runs on a middle layer where it is located the JVM (Java Virtual Machine) responsible for running and managing Java applications (see Fig. 6). Below this layer is the Ginga common Core and the native3D module that is invoked by the native implementation of JSR 239 to render the graphic scenes.

B. Practical Model

In order to validate the architecture designed to Ginga3D a practical model was developed. This practical model implements the proposed functionalities.. In this sense, it was used a reference implementation of Brazilian middleware. The software, called OpenGinga, is an open source implementation developed by researchers in the field of Digital TV. This fact facilitated the access to source code and specification of the system, enabling their study and expansion.

The practical model extends the original modules related to the generation of graphics and processing of user events. Besides, it includes the innovations proposed by Ginga3D to OpenGinga. The execution environment of OpenGinga provides the management and execution of interactive applications, facilitating the creation of testing applications for the Ginga3D. In order to validate the system it was developed interactive 3D applications and performance tests were executed.

C. Case Study: Developing 3D Virtual Environments

It was developed a simplified virtual environment using OpenGL ES in Java (JSR-239) and rebuild the same environment using NCL with X3D/VRML (see Fig. 7). These prototypes were important for analyze certain aspects related to the development of 3D applications in both environments

(imperative and declarative). It was used the same application, so that development process was evaluated on basis of the same requirements. In terms of development were observed the techniques for texturing, collision and event handling, among others.

In terms of architecture the application developed based on Ginga-J (imperative environment) runs on a native implementation of the OpenGL ES API, accessing the features of this native API directly. The application developed with X3D was performed on a simplified X3D browser partially compliant with X3D Interactive Profile specification - Annex C. This browser was implemented on a standard 3D API such as OpenGL.

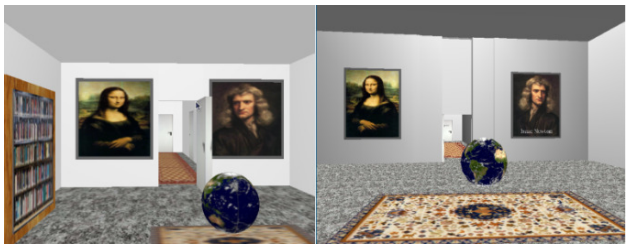

Figure 7. 3D environment developed using OpenGL ES (left) and X3D (right).

The use of browsers avoids the implementation of importers to 3D meshes. However, when using a low level API, like OpenGL ES, the developers need to incorporate, by implementing or using other library, those importers to their applications. The browsers also avoid the need to know deeply about computer graphics techniques. Furthermore, the use of low level APIs allow the construction of customized application using more sophisticated techniques that would not support.

Table 1 shows a comparison between features supported by a graphics API in Java and the language NCL with X3D using a specific browser for rendering. The development of applications using a description language such as X3D or VRML is faster. This is due to the fact that browsers abstract much of the complexity of the computer graphics techniques that are used in these applications. As result, the application development using X3D in the simulated Ginga-NCL environment was 3 times faster than the one developed using a graphics API that could be integrated in the Ginga-J. However, as shown in the Table 1, the use of a graphic library allows greater flexibility in the development of 3D applications since it is possible to choose the techniques of computer graphics to be used. At this point, applications developed with technologies that rely on a browser are restricted to the standard on which it is defined. It was possible to observe that each environment can offer specific benefits depending on the requirements that each application has. Thus, the inclusion of 3D technologies for both environments provides a more flexible approach, leaving developers free to choose the application environment.

TABLE I. COMPARISON BETWEEN THE OPENGL ES API IN JAVA AND NCL WITH X3D.

Characteristic	Ginga-J (Java) JSR 239	Ginga-NCL (NCL+X3D)
Collision Handling	Implemented by the developer – customizable	Supported in the browser - Predefined
Event Handling	Implemented by the developer - customizable	Supported in the browser - predefined
Texturing	Implemented by the developer	Supported in the browser
Lighting Techniques	Fully supported	Partially supported
Robust techniques (Ray trace, Fog, Bump Mapping, etc)	Partially or not supported	Partially or not supported
Time to develop	~3 days	~1 day

VII. RESULTS

In order to test the developed components have been built some interactive 3D applications for the Ginga-J and the Ginga-NCL. Applications for the Ginga-J explore the set of functions contained in the JSR 239 library for the purpose of validating the implementation of this API on the Native3D component. Applications developed to Ginga-NCL environment assess the functionality implemented by the 3D player for X3D documents. For this, it was developed two environments that offer the possibility of navigation and use X3D nodes that were incorporated into the X3D browser also developed in this work.

One of the applications developed using the OpenGL ES API is a simplified animation with three cubes that perform a random motion within the three-dimensional environment (see Fig. 8). The application is built in a graphics area over the video surface, in accordance with the scheme of overlapping surfaces defined by Brazilian standards. For this example were used some of the functions responsible for processing the lighting model, textures and application of material properties on the graphical objects. It was also tested the functions for manipulating OpenGL transformation matrices as well as functions to generate projections. The application runs with a frame rate about 146 fps (frames per second).

The applications developed to the Ginga-NCL environment focused on the presentation of 3D objects using the media player built on the X3D browser. The main goal was to evaluate the 3D interface of the player and their behavior when asked a certain action (play, pause, resume and destroy). One of the applications developed for this environment used an X3D document that describes a small virtual environment.

This document displays a Chinese temple surrounded by a wooden fence (Fig. 9).

Figure 8. JSR 239 application running on middleware Ginga.

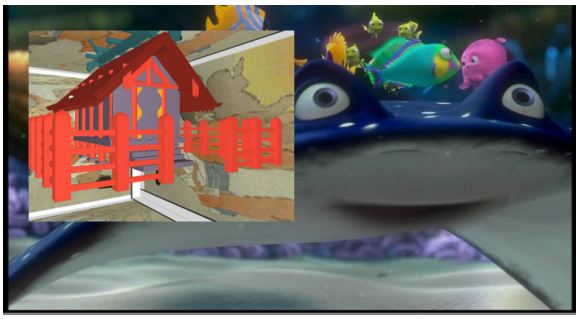

Figure 9. A NCL + X3D application running on the middleware Ginga.

In this paper we have used all nodes implemented by X3D browser. The appearance of the environment was described using the Material node, the objects were described using the IndexFaceSetObject node. The temple was divided into smaller environments, which were integrated in the same scene using the X3D inline node. Other nodes for application of texture, addition of predefined objects and multiple points of light were also used. In this application the frame rate observed was around 58 fps.

A. *Performance Analysis*

In order to analyze the performance of OpenGinga after inclusion of the components of Ginga3D some tests were performed. These tests were carried out to analyze certain aspects such as frame rate of 3D scenes, memory consumption and CPU usage. The first test evaluated the refresh rate of an application by varying the number of primitives that formed the scene. In both environments, the refresh rate refers to the number of frames generated in one second. Fig. 10 shows the variation of the refresh rate for each environment according to the variation in the number of primitives.

In both environments a small number of primitive exhibit high refresh rates. The X3D browser displays frame rates higher than the JSR 239. This is due to the need of execute Java application on a Java Virtual Machine (JVM). When the number of primitives increases, both environments decline in performance. Another fact is that as the number of primitives increases the Java application gain in performance if compared to X3D browser, but for very high values both have similar performance. Up to 20,000 primitive both environments have acceptable frame rates of approximately 30 fps.

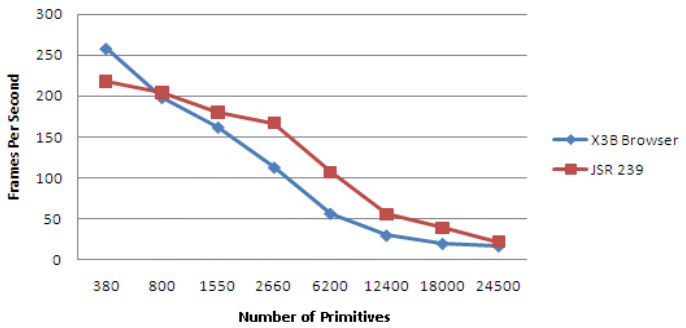

Figure 10. Variation of frame rate for each environment according to the variation of graphical primitives.

A comparative chart for the CPU usage (%) is observed in Fig 11. The chart shows the values of CPU usage for the basic services of Ginga, the JSR 239 and X3D browser. Basic services of Ginga, as video decoding, information system and tuner use about 75% of CPU to perform their basic tasks. Within these services the video decoder is the most critical. On Ginga3D services, when the X3D browser is invoked, the CPU usage is about 83%, since the rendering engine and event handle is activated by browser. When a Java application is executed CPU usage is about 87%. In fact, both environments use the same native component for rendering the scenes. What differentiates them is the need of JSR 239 API invoke a Virtual Machine and use JNI (Java Native Interfaces) JNI to access the functionality of the Native3D component while the X3D browser does this natively.

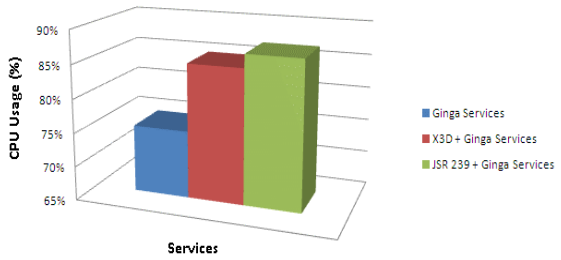

Figure 11. CPU usage for Ginga specific services.

Another aspect evaluated was the memory consumption of basic services of middleware and the services offered by Ginga3D (see Fig 12). The basic services of Ginga consume about 60 Mb of memory to perform their tasks. In case of X3D browser, when running a simple environment, the memory consumption raise to 68.17 Mb. In case of Java this consumption increases to approximately 74.3 Mb The reason for this as in other cases is due to use of virtual machine for running 3D applications.

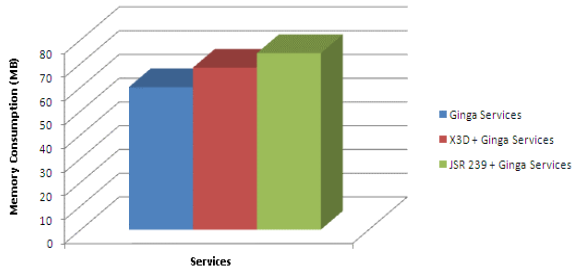

Figure 12. Memory consumption for Ginga specific services.

VIII. CONCLUSIONS

The digitization of media acted directly on the delivery chain of TV productions. This digitization enables the convergence of applications that are transmitted digitally by broadcasting and Internet services in interactive multimedia services. This convergence associated to TV audience allows these interactive services to reach a significant portion of the population. Some examples of these new services are personalized news, program guides, interactive games and applications previously available only in Internet services such as e-mail, home banking, home-shopping and e-learning.

The reducing of computational cost of certain graphics APIs and the upgrade of the graphics hardware for embedded systems contribute to integrate 3D systems into devices with low processing power. These factors allow the use of techniques for generating 3D scenes in a DTV environment, providing a set of new possibilities and ideas with regard to interactive applications.

In this sense, it was presented the Ginga3D, an architecture based on middleware Ginga. Its goal is to extend the Ginga to enable the development and implementation of three-dimensional applications in a DTV environment. The Ginga3D explores the possibilities of using the imperative and declarative environment of middleware to support such applications. As the main difference to other related works, the Ginga3D aims to offer a flexible environment for developing and running applications, so the developer can choose the benefits from the imperative or declarative environment. In this context, another difference of this work is the support to description and synchronization of 3D media through NCL language.

In order to analyze the characteristics of imperative and declarative environment was built a virtual environment using the two strategies. The advantages and disadvantages of each environment were analyzed. Based on this it was concluded that an integration strategy that incorporates the Ginga-J and Ginga-NCL could offer flexibility to developers of three-dimensional applications.

Nowadays, the specification of Ginga3D is completed. Additionally, all components were specified and most of the requirements were finished. Currently, the reference implementation is being concluded to validate the requirements in a real environment. The environment used for testing the reference model use the description of a high-end interactive profile given by Cesar (2005) [34] to execute 3D applications in a DTV environment. In this sense, it is coherent to believe that in the next years, with the decreasing prices of technologies for embedded systems, new set top boxes with more sophisticated hardware requirements will be available and, consequently, will facilitate the implementation of more robust 3D applications.

ACKNOWLEDGMENT

This work was supported by CAPES - Brazilian Federal Agency for Support and Evaluation of Graduate Education.

REFERENCES

[1] G. Olaizola, I. B. Martirena, T. D. Kammann, “Interactive Augmented Reality in Digital Broadcasting Environment”. In Proceeding of 4th European Interactive TV Conference (EuroITV), Athena, Greece, 2006.

[2] L.S. Machado and R.M. Moraes, “Teaching Human Body Structures Using Virtual Reality”. In: World Congress on Engineering and Technology Education Proc. pp. 153-156. 2004.

[3] G. C. Burdea, P. Coiffet, Virtual Reality Technology. New York, Editora John Wiley & Sons, 2003.

[4] Brasil. Ato Presidencial 4091/2006: Instituição do SBTVD, 26 de Novembro de 2003. Brasil. Disponível online em http://www.planalto.gov.br. Último acesso em 17/08/2009.

[5] G. Jones, J. M. Defilippis, H. Hoffmann, E. Williams, A. “Digital Television Station and Network Implementation”. In: Global digital television: Technology & Emerging Services Proceeding of the IEEE, vol 94, nº 01, 2006.

[6] S. Morris, A. Smith-Chaigneau, Interactive TV Standards – A Guide to MHP, OCAP and JavaTV. Burlington, MA, USA, Elsevier, Focal Press, 2005.

[7] ABNT NBR 15601. Televisão digital terrestre — Sistema de transmissão.Technical Report, 2008.

[8] V. Becker, Concepção e Desenvolvimento de Aplicações Interativas para Televisão Digital. Dissertação de Mestrado, UFSC, Florianópolis, 2006.

[9] E. G. Veiga, Modelo de Processo de Desenvolvimento de Programas para TV Digital e Interativa. Dissertação de Mestrado, UNIFACS. 2006.

[10] L. F. G. Soares, “The Multiple Possibilities of Ginga. Professional Production”. Journal on Communication and Audiovisual Techniques, São Paulo, pp. 76 - 83, 2008.

[11] ITU Recommendation J.200. Worldwide common core – Application environment for digital interactive television services. 2001

[12] L. F. G. Soares, “MAESTRO: The Declarative Middleware Proposal for the SBTVD”. Proceedings of the 4th European Interactive TV Conference. Athena, 2006.

[13] L. E. C. Leite, C. E. C. F. Batista, G. L. Souza Filho, R. Kulesza, L. G. P. Alves, G. Bressan, R. F. Rodrigues, L. F. G. Soares, “FlexTV – A Proposal for a middleware architecture for the Brazilian System of Digital Television”. In Jounal of Computer Engineering and Digital Systems, São Paulo , v. 2, pp 29-50, 2005.

[14] L. F. G. Soares, R. F. Rodrigues, M. F. Moreno, “Ginga-NCL: the Declarative Environment of the Brazilian Digital TV System”. In: Journal of the Brazilian Computer Society. Porto Alegre, RS, n. 4, Vol. 13. pp.37-46. ISSN: 0104-6500, 2007.

[15] M. J. Antonacci, NCL: A Declarative Language for Specification of Hypermedia Documents. Master Dissertation. Department of Computer Science, PUC-Rio, 2000.

[16] ECMA-262. ECMAScript Language Specification. Available in: <http://www.ecma-international.org>. Last access: August, 2009

[17] G. L. Souza Filho, L. E. C. Leite, C. E. C. F. Batista, “Ginga-J: The Procedural Middleware for the Brazilian Digital TV System”. In: Journal

of the Brazilian Computer Society. Porto Alegre, RS, nº. 4, Vol. 13. pp.47-56. ISSN: 0104-6500, 2007.

[18] L. D. N. A. Silva, Proposal API to develop applications for Multi-User and Multi-Device for DTV using the Middleware Ginga. Master Dissertation, Department of Computer Science, João Pessoa, UFPB, 2008.

[19] J. D. Foley, A. V. Dam, S. K. Feiner, J. F. Huges, Computer Graphics: Principles and Practice. Boston, Addison Wesley, 1997.

[20] T. A. Möller, E. Haines, Real-Time Rendering. Massachusetts, A.K. Peters, 2002.

[21] A. Watt, 3D Computer Graphics. England, Addison Wesley, 2000.

[22] T. Mcreynolds, D. Blythe, Advanced Graphics Programming Using OpenGL. San Francisco, Elsevier, 2005.

[23] D. Hearn, M. P. Baker, Computer Graphics with OpenGL. Upper Saddle River, Prentice Hall, 2004.

[24] D. Brutzman,, L. Daly, X3D – Extensible 3D Graphics for Web Authors. Elsevier, 1st Ed, San Francisco, 2007.

[25] S. Maad, "Universal Access For Multimodal ITV Content: Challenges and Prospects", In: 7 th ERCIM Workshop on User Interfaces for All, Paris, France. SPRINGER LNCS proceedings of the ERCIM, 2002.

[26] S. Maad, "The potential and pitfall of interactive TV technology: an empirical study". In: International Conference on Television in Transition, England, 2003.

[27] S. Maad, "MARILYN: a novel platform for intelligent interactive TV (IITV)". In Jacko, J.A., Stephanidis, C. (Org.),Human-Computer Interaction, Theory and Practice (Part 1), Lawrence Erlbaum Associates, Mahwah, NJ, Vol. 1 pp.1041-5, 2003.

[28] T. D. Kammann, Interactive Augmented Reality in Digital Broadcasting Environments. Master Dissertation. University of Koblenz and Landau, Koblenz, 2005.

[29] C. Fehn, R. Barre, S. Pastoor, "Interactive 3-DTV-concepts and key technologies". Proceeding of. IEEE, vol. 94, n 3 pp. 524-538, 2006.

[30] J. Segnès, Y. Fisher, A. Eleftheriadis, "MPEG-4's binary format for scene description". In: Signal Processing: Image Communication, vol. 15, pp. 321-345, 2000.

[31] R. Pulles, P. Sasno, "A set top box combining MHP and MPEG-4 interactivity". In: 2nd European Union Symposium on Ambient Intelligence. Eindhoven, The Netherlands, Proceedings of the 2nd European Union Symposium on Ambient Intelligence, pp. 31-34, 2004.

[32] E. Boyle, "The creation of MPEG-4 content and its delivery over DVB infrastructure". In Proceedings of the first Joint IEI/IEEE Symposium on Telecommunications Systems Research, Dublin, Ireland, 2001.

[33] K. Illgner, J. Cosmas, "System concept for interactive broadcasting consumer terminals". In: Proceedings of the International Broadcasting Convention (IBC), Amsterdam, The Netherlands, 2001.

[34] P. Cesar,. A Graphics Software Architecture for High-End Interactive TV Terminals. Doctoral Thesis, Helsinki University of Technology, Helsinki, 2005.

[35] P. Cesar, P. Vuorimaa, J. Vierinen, "Graphics architecture for high-end interactive television terminals". ACM Transactions on Multimedia Computing, Communications, and Applications, vol. 2, pp. 343-357, 2006.

[36] ABNT NBR 15606-2, Televisão digital terrestre – Codificação de dados e especificações de transmissão para radiodifusão digital Parte 2: Ginga-NCL para receptores fixos e móveis – Linguagem de aplicação XML para codificação de aplicações. 2008.