Suturas en Medicina Veterinaria: Un Prototipo de Desarrollo NCL para la plataforma de Televisión Digital

Martinelli, Fernán Gabriel
Riberi, Franco Gaspar
Varea, Agustín

Director: Arroyo, Marcelo

Co-Director: Aguirre, Nazareno

Universidad Nacional Río Cuarto

04/07/2011

Resumen

Se presentará una aplicación para la plataforma de televisión digital. Esta aplicación está integrada a un video educativo sobre Medicina Veterinaria, y cumple la función de proveer información adicional sobre el tema abordado en el video. Esta información se presenta opcionalmente (el usuario decide si visualizarla o no) de forma similar a una aplicación web, brindando al usuario la posibilidad de elegir a través de menúes interactivos la información deseada.

La aplicación está desarrollada sobre la plataforma middleware GINGA, y combina características de los lenguajes Nested Context Language NCL (declarativo) y LUA (imperativo), para conseguir interactividad con el usuario. Esta interactividad incluye funcionalidades relativamente simples, como la provisión de diferentes tipos de información en momentos distintos del video, y algunas más elaboradas, como la posibilidad de realizar actividades de autoevaluación (tests, preguntas y respuestas, etcétera). La aplicación fue diseñada teniendo en cuenta su portabilidad, y puede ejecutarse/visualizarse a través de su transmisión por TV digital, o alternativamente como una aplicación *stand-alone*, provista en un live CD.

Palabras clave: TV Digital, NCL, Lua, GINGA.

1. Introducción

La televisión es un medio de comunicación que actualmente se encuentra en constante crecimiento, existe un cambio masivo que los diferentes países del mundo están atravezando, este cambio consiste en el salto de televisión analógica a la televisión digital. El impacto de la TV digital es mucho más significativo que una simple mejora en la calidad de imagen y sonido. Más que eso, un sistema de TV Digital permite mayor flexibilidad, facilitando expandir las funcionalidades del sistema mediante aplicaciones, en otras palabras, desarrollo de software para TV Digital. Tales aplicaciones son programas computacionales residentes en dispositivos receptores (conocidos como Set-Top Box) o provenientes de datos enviados conjuntamente con el audio y video principal de un programa televisivo. La integración de una capacidad computacional a los Set-Top Box permite el surgimiento de una amplia gama de servicios y programas de TV compuestos no sólo por el audio y video principal, sino también por otros videos, imagenes, textos, etcétera, sincronizados en una aplicación muchas veces guiada por la interacción del usuario telespectador. El universo de las aplicaciones de TV Digital puede ser particionado en un conjunto de aplicaciones declarativas, desarrolladas en el lenguaje de programación NCL, y un conjunto de aplicaciones imperativas desarrolladas en el lenguaje Lua, que pueden estar relacionadas entre sí. Es necesario que las aplicaciones sean independientes de la plataforma de hardware y software, por lo cual se añade una nueva capa al modelo de referencia del sistema TV Digital, denominada GINGA, este es un middleware que permite ejecutar aplicaciones interactivas dentro de un Set-Top-Box. Dada la innovación que esta tecnología presenta, se optó por el desarrollo de una aplicación para esta plataforma. Tal aplicación tiene como objetivo la digitalización de un video particular, entendiéndose por este término, la agregación de interactividad usando las herramientas provistas por TV Digital. El video sobre el cual fue realizada la aplicación lleva el nombre *Punto por Punto: Suturas*. Esta producción audiovisual fue provista por la Facultad de Veterinaria, más precisamente por el área de Medicina Veterinaria de la Universidad Nacional de Río Cuarto. *Punto por Punto: Sutura* fue elaborado, diseñado e implementado en el marco de un Proyecto de Innovación e Investigación para el mejoramiento de la Enseñanza de Grado (PIMEG) Res Rec 1331/08. Secretaria Académica, Secretaria de Ciencia y Técnica y Secretaria de Planificación y Relaciones Institucionales de la UNRC. El mismo es el primero de una tira de videos educativos, los cuales tienen como objetivo educar a los estudiantes de veterinaria para la realización de las suturas. Éstos son un recurso didáctico como propuesta en la enseñanza de la cirugía veterinaria, debido a la dificultad en los procedimientos para desempeñar las maniobras necesarias para realizar una sutura, falta de material de estudio y principalmente la dificultad en la práctica hospitalaria. Es importante destacar que se trabajó con el Sistema Argentino de TV Digital Terrestre. A continuación se detallan los conceptos mencionados y se presenta dicha aplicación.

2. Televisión Digital

Existen en la actualidad cuatro medios principales para que una señal de televisión llegue a los hogares: Cable, Satelital, IPTV y Terrestre. El más común en

Argentina es el cable, es decir, un tendido de cables lleva la televisión a aquellos que contratan el servicio. De a poco se ha hecho más popular la televisión satelital, donde un proveedor "sube" una señal a un satélite que "al bajar" cubre gran parte del país. Sin embargo, hasta ahora la televisión satelital es un servicio sólo para abonados, lo que significa que aunque la señal llegue a los hogares, sólo puede verse si se cuenta con la antena correspondiente y el decodificador adecuado. Recientemente, se comenzó a hablar de IPTV, o televisión por IP. Se trata de un servicio residencial en el que un proveedor utiliza una red IP para llevar señal de televisión a sus abonados. Finalmente, tenemos la televisión terrestre, también conocida como televisión de aire, donde una antena con forma de parrilla ubicada sobre los hogares capta una señal de VHF[1] que el televisor reproduce. La televisión terrestre es la única abierta de todas las disponibles en el país. Esta forma de televisión se conoce como analógica, esto quiere decir que la señal que se transmite vía VHF, es una representación analógica del video y audio del canal de televisión en cuestión. Es una tendencia mundial migrar desde una transmisión analógica a una digital, la cual permite añadir el envío de datos como característica importante.

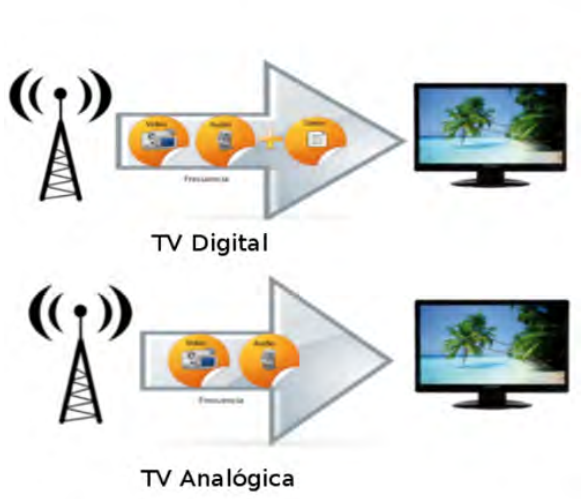


Figura 1: Transmisión en TV analógica y digital

En un sistema de televisión analógica, el canal de transmisión presenta diversas formas de interferencias y ruidos en la señal original, lo que limita la capacidad del sistema. Estos ruidos estan presentes en todo el espectro y no pueden ser evitados, provocando la reducción de la calidad de la señal (aparición de "llovizna" en la imagen). La disminución de la calidad depende de una relación señal/ruido (S/N). Cuando la relación disminuye, se reduce la calidad de señal. En los sistemas digitales, a baja relación de S/N aumenta la probabilidad de error de bit (modificación de la señal). Para evitar este problema, todos los patrones de transmisión utilizan códigos de corrección de errores. Esto es, si la tasa de error está por debajo de un umbral, el código es capaz de corregir

todos los errores introducidos por el canal y no hay pérdida de la calidad de la imagen. Sin embargo, si la tasa de error es superior al umbral, el sistema no será capaz de corregir el código de transmisión. Este es uno de los motivos por los cuales se acostumbra a decir que la televisión digital tiene una señal de imagen perfecta, sin "ruido", o en su defecto, ninguna imagen. El deteriodo también puede ser causado por múltiples vías o superposición de varias señales con diferentes atenuaciones y pequeños retrasos produciendo interferencias, como se muestra en la Figura 2.

Figura 2: Efecto de múltiples rutas

En televisión analógica causa la aparición de "fantasmas". En contrapartida, en televisión digital se puede producir una interferencia entre símbolos, es decir, una superposición entre los bits transmitidos. Si el rango de duración de esta interferencia de símbolos es mayor a un umbral, nuevamente se pierde la imagen (TV Digital es todo o nada). Se dice que los sistemas digitales deben tener la suficiente robustez para evitar éstos problemas.

TV Digital permite mejor imagen y sonido gracias a las técnicas de compresión de datos, lo que permite la producción de señales de alta resolución (señales de mejor calidad). El Sistema Argentino de TV exige que la tasa de bits sea transmitida en un canal de televisión de 6 Mhz. La TV digital permite que en el mismo canal de 6 MHz, se transmita imagen de alta definición llamada HDTV. Las técnicas de compresión digital también permite la alternativa de tener múltiples programas de más baja calidad (SDTV) en lugar de tener un solo programa de alta calidad (HDTV) que ocupan toda la banda de 6 Mhz. Esto es lo que llamamos multiprogramación y puede observarse gráficamente en la Figura 3.

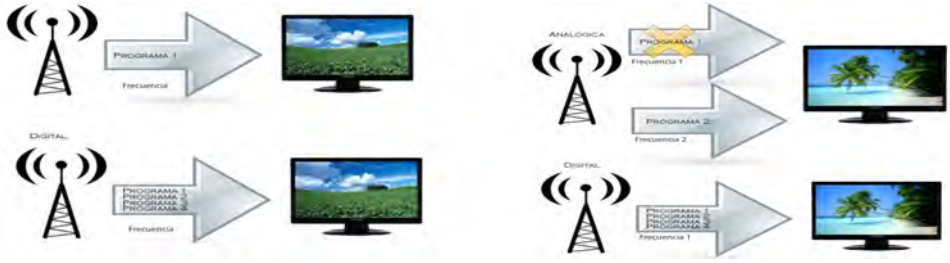


Figura 3: Multiprogramación y cambio de canal

En TV analógica, video, audio e información de datos limitados (como

subtítulos) son transmitidos multiplexados, de modo que un receptor de diseño relativamente simple, puede decodificar y volver a montar los diversos componentes de la señal para producir un programa. Como tal, un programa se transmite en su forma final. Sin embargo, en un sistema de televisión digital se necesitan niveles adicionales de tratamiento. El receptor procesa la señal digital mediante la extracción de la señal recibida de una colección de elementos del programa, como son video y audio principal y datos (éstos pueden ser objetos multimedia (video, sonido, imgenes, texto, etcétera) y también programas NCL y Lua) que conforman el servicio seleccionado por el consumidor y que provienen de lo que se denomina carrusel de datos. Esta selección se realiza mediante el servicio del sistema y la información, que también se transmiten. Dado que los objetos se han recibido, son procesados de acuerdo a una solicitud también recibida junto a los datos, que se encarga de la presentación final. El carrusel de datos, observado en la Figura 4, da soporte al envío cíclico de los datos, dado a que la sincronización de un canal de televisión puede ser realizada en cualquier momento, por lo cual los datos que no tengan relación temporal especificada por medio de etiquetas de tiempo deben ser enviados cíclicamente. Si se hace esto, la recepción de estos datos sería independiente del instante de tiempo. De manera similar, TV digital posee un carrusel de objetos que permite el envío cíclico de un sistema de archivos. Así, para sincronizar un determinado canal, el receptor debe ser capaz de decodificar los datos recibidos y colocarlos en un área de memoria para que puedan ser utilizados, preservando la estructura de archivos y directorios enviada. Por otra parte, las aplicaciones transferidas en archivos de un carrusel pueden hacer referencia a recursos presentes en el mismo carrusel o a recursos de otros carruseles. Por ejemplo, una página HTML transmitida en un carrusel puede referenciar una imagen cuyo contenido es transmitido en otro carrusel de objetos. El carrusel de objetos es en efecto un protocolo de transmisión cíclica de datos. El resultado es un flujo de datos que contiene el sistema de archivos transmitido de forma cíclica. Así, si un determinado receptor no ha recibido un bloque de datos en particular (debido a un fallo en la transmisión ó por haber sintonizado el canal después de la transmisión de ese bloque), basta esperar por su retransmisión correcta.

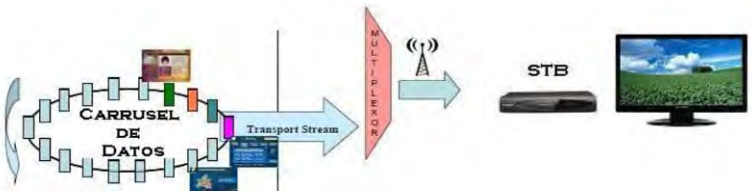


Figura 4: Carrusel de datos

La capacidad de cálculo necesaria para el nuevo sistema puede ser integrada en el dispositivo de visualización, algunos de ellos pueden ser por ejemplo un televisior, un teléfono, un PDA, etcétera. Debido al gran legado de aparatos de televisión analógica, una solución simple para ofrecer estos nuevos servicios a los mencionados sistemas receptores es utilizar junto a la televisión, un "sistema de procesamiento" capaz de manejar correctamente la señal de radiodifusión, que decodificará y mostrará en la televisión consistente, la programación, aplica-

ciones y servicios avanzados. Este sistema de tratamiento consiste en una caja convertidora de la señal digital (o set-top box). La señal recibida después de demodulada y separada (demultiplexada), se entrega a los decodificadores de audio y video, y procesamiento de CPU. El receptor tiene acceso a otra red (red externa en la Figura 5) a través del cual se puede recibir o enviar datos según lo ordenado por las solicitudes que reciba. El canal de acceso a la red se llama canal de retorno o devolución. En un sistema de IPTV o P2PTV, por lo general la señal no es modulada, va directamente a un módulo demultiplexor; Por otra parte, también es habitual que el canal interactivo esté formado por la misma red de transmisión de donde proviene la transmisión de datos multiplexados. Un sistema de televisión digital es un sistema típico cliente/servidor. El entorno de servidor consiste en un organismo de radiodifusión (parte izquierda de la Figura 5) o un proveedor de contenido y el cliente, el visor de entorno de usuario (parte derecha de la Figura 5).

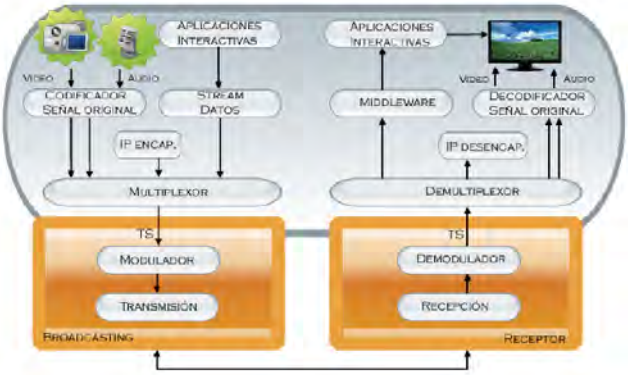


Figura 5: Sistema de TV Digital

Un programa se compone de un video principal y un audio principal, capturado en vivo desde una cámara, o de un servidor de video. El mismo, junto al audio, se entregan a los codificadores digitales, que son responsables de generar respectivos flujos de video principal y audio principal comprimidos. Estos flujos, son multiplexados en una sola señal, llamada flujo de transporte (TS - Transport Stream).

La señal es recibida, demodulada y entregada al demultiplexador, el cual separa los principales flujos de video y audio principal de los flujos de datos, que se entregan para su procesamiento. El tratamiento de los datos recibidos pueden requerir nuevos datos, obtenidos a partir del canal interactivo o canal de retorno. Es importante destacar, que un sistema de TV Digital puede funcionar sin canal de retorno (o canal de interactividad). En este caso, las aplicaciones pueden utilizar sólo los datos recibidos desde el emisor. Cuando las aplicaciones permitan la interacción del usuario, el servicio ofrecido se denomina de Interactividad Local. Sin embargo, la TV Digital puede incluir el uso un canal de

retorno. Éste puede ser unidireccional, sólo permitiendo que el receptor envíe datos. Un segundo nivel de interactividad se define entonces, permitiendo al usuario telespectador el envío de datos, por ejemplo, su interés en la compra de un determinado producto, la votación en un tema en particular, etcétera. El canal de retorno también puede ser bidireccional, posibilitando al receptor descargar los datos utilizados por las aplicaciones. En este caso, una aplicación puede recibir datos por difusión o por la red de retorno. También es posible un tercer nivel de interactividad, lo que permite el acceso a los datos no provenientes de la emisora, por ejemplo, lo que permite navegar por la web. Un canal de retorno bidireccional también puede permitir el envío de datos a través de banda ancha. En este caso, el receptor puede empezar a actuar como una pequeña estación.

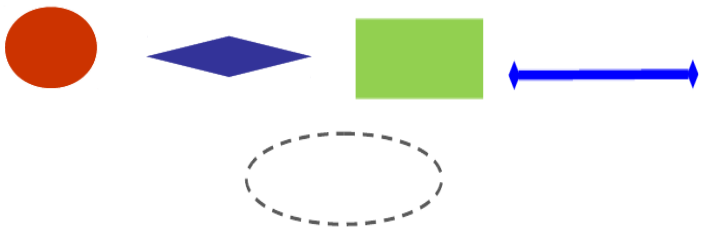
3. Nested Context Languaje (NCL)

NCL[3] es un lenguaje declarativo, el cual separa los contenidos media de la estructura de la aplicación. El lenguaje está basado en un modelo conceptual de datos llamado Nested Context Model (NCM)[2]. Este modelo conceptual permite la representación de elementos multimediales, sincronizarlos en tiempo y espacio para crear aplicaciones interactivas.

Un documento NCL define cómo los objetos media son estructurados y relacionados, tanto en tiempo como en espacio.

Las entidades básicas que permite representar el lenguaje NCL son:

- Objetos Media - Descriptor - Región - Link



- Contexto

Un objeto media puede ser un video, audio, texto, html, lua, o una imagen. En el apéndice A se muestran los distintos formatos soportados. Un descriptor indica en que región se mostrará el objeto media y con que propiedades. Una región representa un área de un dispositivo en la cual ciertos objetos media serán visualizados. Gráficamente puede observarse esta representación en la Figura 6.

Figura 6: Asociación entre media, descriptor y región

Los link permiten especificar acciones sobre objetos media a partir de la ocurrencia de eventos, por ejemplo:

Condición: "al presionar" el boton amarillo" **Acción**: "mostrar una imagen"

Un contexto permite agrupar elementos NCL y reutilizarlos.

3.0.1. Sintaxis del Lenguaje

La estructura general de un documento NCL es la siguiente:

```
<?xml version=‘‘1.0’’ encoding=‘‘ISO-8859-1’’?>          (1)
<ncl id=‘‘main’’ xmlns=‘‘http://www.ncl.org.br/NCL3.0/EDTVProfile’>    (2)
    <head>
        region, descriptor, connector
    </head>
    <body>
        port, media, switch, link
    </body>
</ncl>
```

(1) Este es un encabezado XML, el cual debe estar presente en todo documento NCL.

(2) El elemento ncl posee dos atributos un id y xmlns, que identifican la aplicación y el perfil del lenguaje utilizado.

Estructura del head

```
<head>
    <importedDocumentBase>
        <!--aplicaciones NCL importadas y referenciadas-->
    </importedDocumentBase>
    <ruleBase>
        <!--reglas para aceptar el contenido y su forma de presentacion-->
    </ruleBase>
    <transitionBase>
        <!--efectos de transicion para la presentacion de objetos-->
    </transitionBase>
    <regionBase>
        <!--Areas de exhibicion destinadas a los objetos de media-->
    </regionBase>
    <descriptorBase>
        <!--configuraciones de presentacion de objetos media-->
    </descriptorBase>
    <connectorBase>
        <!--comportamiento de los objetos-->
```

```
    </connectorBase>
    <meta/> <!--datos descriptivos simples-->
    <metadata>
        <!--datos descriptivos estructurados-->
    </metadata>
</head>
```

Estructura del body

```
<body>
    <port .../>
    <media .../>
    <media>
        <area .../>
    </media>
    <context>
        ...
    </context>
    <switch>
        ...
    </switch>
    <link>
        <bind ... />
    </link>
</body>
```

3.0.2. Semántica del Lenguaje

En el apéndice B se detalla la semántica del lenguaje.

4. LUA

Lua[4] es un lenguaje de programación imperativo, estructurado, bastante ligero y libre. La semántica de Lua puede ser extendida y modificada redefiniendo funciones de las estructuras de datos utilizando metatablas. Lua ofrece soporte para funciones de alto orden y recolector de basura. Combinando todo lo anterior, es posible utilizar Lua en programación orientada a objetos.

Al ser un lenguaje de extensión, Lua no tiene noción de "programa principal", sólo funciona integrado en un cliente de acogida, se llama programa embebido, o simplemente el de acogida. Las variables no tienen tipo, sólo los datos y pueden ser lógicos, enteros, números con punto flotante o cadenas. Las estructuras de datos como matrices, conjuntos, tablas hash, listas y registros pueden ser representados utilizando la única estructura de datos de Lua: tabla.

Lua es normalmente usada para permitir que una aplicación principal sea extendida o adaptada a través del uso de scripts. La aplicación principal puede ser un video juego, donde un script Lua puede ser usado para definir el comportamiento de un personaje, o un editor de textos, que permite que los textos sean accedidos y modificados por scripts Lua, ó de manera más general, aplicaciones

que usan Lua en scripts de configuración (este tipo de uso caracteriza Lua como un lenguaje de scripts).

Lua es un lenguaje de fácil aprendizaje, que combina sintaxis procedural con declarativa, con pocos comandos primitivos. De esa característica resulta una implementación eficiente comparada con la de otros lenguajes de característica similares. Lua también presenta alto grado de portabilidad, pudiendo ser ejecutado con todas sus funcionalidades en diversas plataformas, tales como computadoras personales, celulares, sistemas embebidos, entre otros.

Una explicación más detallada de la sintaxis y funcionalidades de Lua escapan a este trabajo, por lo cual explicaremos los conceptos del lenguaje necesarios para aplicaciones de TV Digital.

Para adecuar el ambiente de TV Digital e integrarse con NCL, el lenguaje Lua fue extendido con nuevas funcionalidades. Por ejemplo, un objeto NCLua presisa comunicarse con el documento NCL para saber cuando el objeto <media> correspondiente es iniciado por un link. Un NCLua también puede responder a teclas del control remoto. La diferencia entre un NCLua y un programa Lua puro es el hecho de ser controlado por un documento NCL.

Algunas bibliotecas proporcionadas por Lua para scripts NCLua son:

Módulo event: permite que objetos NCLua se comuniquen con el documento NCL y otras entidades externas (tales como control remoto y canal de interactividad).

Módulo canvas: ofrece funcionalidades para diseñar objetos gráficos en una región de NCLua.

Módulo settings: ofrece acceso a las varibles definidas en el objeto settings de un documento NCL (objeto de tipo "application/x-ncl-settings").

Módulo persistent: exporta una tabla con variables persistentes entre ejecuciones de objetos imperativos.

Los mecanismos de integración de objetos NCLua con documentos NCL tienen sustento en el paradigma de programación orientada a eventos.

No sólo la comunicación con el NCL, sino también cualquier interacción con aplicaciones externas tales como canal interactivo, control remoto y el temporizador, tienen sustento en la difusión y recepción de eventos. El módulo *event* de NCLua utilizado para este propósito es la extensión más importante de Lua, y su comprensión es esencial para el desarrollo de cualquier aplicación que utiliza objetos NCLua. En el apéndice C se aborda con más detalle conceptos Lua.

5. Middleware GINGA

El sistema de TV Digital utiliza un middleware que permite ejecutar aplicaciones interactivas dentro de un Set-Top-Box (Figura 7).

Una aplicación puede realizarse directamente en el sistema operativo de un receptor. Sin embargo, los sistemas operativos en general no están preparados para dar un buen soporte a las aplicaciones de la televisión digital. Además, una aplicación de televisión debe ser capaz de funcionar en cualquier plataforma hardware y software. Para hacer aplicaciones independientes de plataforma de hardware y software, una nueva capa se añade a los puntos de referencia de un sistema TV digital. Esta capa es denominada GINGA. El middleware es uno de los componentes más importantes de un sistema de televisión digital porque en la práctica, es lo que rige las relaciones entre dos industrias de importancia: la

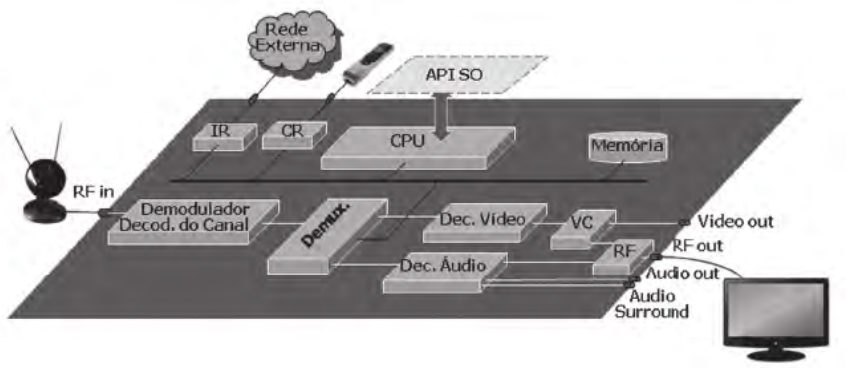


Figura 7: Estructura

producción de contenidos y la fabricación de receptores. El middleware abierto GINGA se subdivide en dos subsistemas principales interrelacionados, que permiten el desarrollo de aplicaciones siguiendo dos paradigmas de programación diferentes. Dependiendo de las funcionalidades requeridas en cada proyecto de aplicación, un paradigma será más adecuado que otro. Estos dos subsistemas se llaman Ginga-J [5] (para aplicaciones procedurales Java) y Ginga-NCL[6] (para aplicaciones declarativas NCL). Ginga-J provee una infraestructura de ejecución de aplicaciones Java y extensiones específicas hacia el ambiente de TV. Ginga-NCL es un ambiente de presentación multimedia para aplicaciones declarativas escritas en NCL y usa un lenguaje de script llamado LUA. Hoy en día, Ginga-NCL es el único estándar internacional tanto para IPTV como para TV digital terrestre. Fue creado en la Pontificia Universidade Católica de Río de Janeiro (PUC-Rio) y ofrece una infraestructura de presentación de aplicaciones de multimedia/hipermedia. GINGA tiene una especificación abierta, es una implementación de referencia que se ha liberado bajo licencia GPL.

6. Aplicación

6.1. Objetivo

Esta aplicación tiene por objetivo, proveer un prototipo de aplicación NCL-LUA que permita la digitalización de un video particular(en este caso del género educativo), entendiendose por tal, a la posibilidad de agregar interactividad mediante el uso de las herramientas que posibilita la TV Digital. Se plantea definir un modelo que respete las buenas prácticas de programación, y que permita tanto a los desarrolladores como a los telespectadores obtener parámetros sobre los alcances de la TV Digital.

6.2. Descripción

La aplicación permite al telespectador visualizar, con los 4 botones de interactividad Ginga, información adicional relevante al video-documental ordenada mediante un menú. Estos botones son un estándar en el control remoto del Set-

Top Box de TV Digital, los cuales poseen distintos colores: rojo, verde, amarillo y azul. El middleware GINGA en la PC, permite emular estos botones mediante las teclas F1, F2, F3 y F4 respectivamente. Retomando al menú, cada opción del mismo está relacionada a un color de estos botones. El telespectador puede navegar a través de los menúes que propone la aplicacón utilizando estos botones de color y en algunos casos, cuando se lo solicite, el mismo deberá utilizar las flechas de navegación del control remoto. Mientras el telespectador navega, el prototipo va presentando diferentes maneras de exhibir la información. Una de ellas es mostrar imagenes y la otra es en forma de texto. La aplicación también provee una funcionalidad de autoevaluación, la cual llamamos "test points", que consiste en: al finalizar cada capítulo del video, se le ofrece al usuario la opción de contestar una pregunta de tipo múltiple choice. Las opciones poseen alguno de los colores, antes mencionados, relacionado a la misma. Luego de efectuar la elección, la aplicación mostrará un tilde en caso de ser correcto o una cruz en caso contrario. Ésta es recordada y al finalizar el video se le informará la cantidad de respuestas correctas. El telespectador puede optar por ignorar el test. Es importante destacar que en ningún momento de la aplicación se pierde de vista el video-documental. Se trató de no quitar demasiado la atención del telespectador sobre el video por cuestiones pedagógicas. Por último, la aplicación fue embebida en un Live CD, el cual lanza la aplicación GINGA de forma automática, dado que no se cuenta aún con un posible medio para la transmisión de manera digital por aire ante una emisora (exceptuando a Buenos Aires).

6.3. Diseño

En el apéndice D se detalla como se modelo la aplicación.

6.4. Funcionamiento

En la Figura 8 se exhibe el comienzo de la aplicación, donde el video principal es lanzado junto a un botón de interactividad opcional. Este botón indicador de interactividad GINGA permanece activo durante todo el video, a excepción, de los instantes de tiempos en que se exhibe la posibilidad de realizar los test.

Figura 8: Inicio de la aplicación

El telespectador tiene total control sobre lo que se exhibe en la pantalla, es decir, recae en él la decisión de visualizar la información extra que la aplicación

brinda. En caso de que el mismo decida ver dicha información debe presionar el botón menú del control remoto ó en su defecto la tecla F5 del teclado dependiendo del dispositivo utilizado. Efectuada esta acción, se lanza un menú el cual administra las distintas opciones de información disponible.

Figura 9: Menú inicial

Como se puede observar en la Figura 9, se presentan 4 alternativas para el usuario. Una de ellas, la más simple (*volver*), proporciona la opción para volver a visualizar el video en toda la pantalla como se observó en la Figura 8. Las opciones *sinopsis*, *instrumental* e *información anexa* agregan información sin pasar a otro estado, es decir, el video no se redimenciona, sino que la información aparece directamente en la parte inferior. La opción sinopsis, Figura 10, presenta datos interesantes a cerca del video, datos de sus autores, contenidos, etcétera. Estos datos se visualizan en forma de texto. Para una perfecta apreciación del contenido, se presenta la información por partes con un tamaño adecuado, donde el telespectador puede acceder a más información a través de las flechas de navegación. La opción instrumental, Figura 11, consta a su vez de un segundo menú, por lo cual, el menú incial es deshabilitado opacándose teniendo el control momentaneo el menú secundario. Éste último, permite acceder a una gran cantidad de información, y su tratamiento es muy similar al menú inicial. Se destaca la opción *instrumental gral*, Figura 12, la cual brinda una amplia cantidad de imagenes que exiben herramientas de trabajo importantes con su correspondiente descripción en forma de texto. La opción *info anexa* tiene un tratamiento similar a sinopsis, con la diferencia que la información se presenta en un cuadro.

Cabe reiterar, dada su importancia, que en todo momento el usuario es quien decide qué información se exhibe. Como se mencionó anteriormente, es posible la autoevaluación acerca del video. Para indicar que se encuentra disponible un test, se muestra un botón, si el usuario accede, se visualiza una pregunta la cual puede responderse, o ignorarse, como se muestra en la Figura 13. El test tiene un tiempo determinado, por lo cual, pasado ese tiempo, el test desaparece. Esta es una decisión de implementación, que pretende que el telespectador no se pierda detalles del video.

Figura 10: Resultado de seleccionar la opción Sinopsis

Figura 11: Resultado de seleccionar la opción instrumental

7. Conclusión

En este trabajo, hemos presentado el desarrollo de una aplicación para la plataforma de TV Digital. Esta aplicación combina el uso de dos lenguajes, NCL para la descripción de la integración entre datos, video y sonido (y su respectiva sincronización), y LUA para la descripción de aspectos más algorítmicos vinculados a la aplicación de TV digital. El programa de TV desarrollado, si bien es un prototipo, hace uso de muchos aspectos avanzados de la plataforma de desarrollo elegida, aprovechando entre otras características la separación adecuada entre hardware y software que la plataforma GINGA provee. El diseño presentado es en cierto sentido *portable*, desde el punto de vista que la aplicación puede transmitirse a través de TV digital terrestre, o ejecutarse como una aplicación educativa *stand alone*. Por lo tanto, consideramos a esta propuesta una primera experiencia en una plataforma interesante para el desarrollo de material educativo, entendiendose a material la combinación de información multimedial (audio, video), y *software*.

Dado el significativo impacto que la TV digital está teniendo en nuestro país, particularmente debido al alcance que la misma logrará en todo el territorio y las posibilidades que brindará para la difusión de contenidos, creemos que nuestra experiencia en este desarrollo constituye un punto de partida para la posibilidad

Figura 12: Resultado de seleccionar la opción instrumental gral de menú proporcionado por la opción instrumental

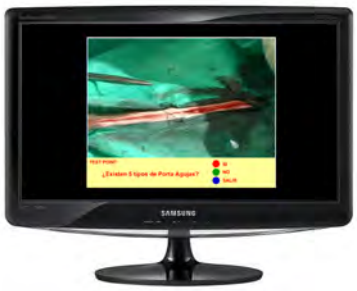


Figura 13: Resultado de acceder al test point

de desarrollos basados en esta plataforma desde la Universidad Nacional de Río Cuarto. La rapidez con la que se espera tener el sistema de TV digital terrestre en completo funcionamiento nos impondrá considerables demandas en la generación de contenidos, y la adaptación o extensión de contenidos multimedia existentes con capacidades de interacción brindados por la plataforma. Nuestro desarrollo ha dado lugar a la interacción con un equipo de trabajo en nuestra Institución que favorecerá este tipo de desarrollos.

Entre los trabajos futuros, estamos trabajando en la adaptación de otros productos desarrollados por el departamento de Audiovisuales de la UNRC, con características similares a las mostradas en la aplicación presentada aquí. También estamos trabajando en la definición de plantillas genéricas que faciliten el uso de esta plataforma para el desarrollo de software educativo que abarque contenidos multimediales.

8. Referencias

[1] Link señal VHF television analógica
[2] Capítulo 2. PROGRAMANDO EM NCL. Luiz Fernando Gomes Soares, Simone Diniz Junqueira Barbosa. 2009. Elsevier Editoria Ltda. ISBN: 978-85-352-

3457-2.
[3] Capítulos 3-15. PROGRAMANDO EM NCL. Luiz Fernando Gomes Soares, Simone Diniz Junqueira Barbosa. 2009. Elsevier Editoria Ltda. ISBN: 978-85-352-3457-2.
[4] Capítulo 18. PROGRAMANDO EM NCL. Luiz Fernando Gomes Soares, Simone Diniz Junqueira Barbosa. 2009. Elsevier Editoria Ltda. ISBN: 978-85-352-3457-2.
[5] referencia asociada a GINGA-jv
[6] http://www.gingancl.org.br/
http://comunidad.ginga.org.ar/ // http://gingarn.wikidot.com/

A. Formatos de Medios Soportados

Los formatos de media soportados en TV Digital se exhiben en la siguiente tabla.

Tipo de media	Extension
IMAGEN	bmp png gif jpg jpeg
AUDIO	wav mp3 mp2 mpeg mpg mp4 mpg4
VIDEO	mpeg mpeg4
TEXTO	htm html txt css xml

Figura 14: Formatos de TV Digital

B. El Lenguaje NCL

Un documento NCL está compuesto principalmente por dos partes <*head*> y <*body*>. El elemento <head> contiene las bases de los elementos referenciados por el núcleo de la aplicación NCL, como regiones, descriptores, transiciones, entre otros. El elemento <body> contiene los elementos que definen el contenido de la aplicación propiamente dicha, tal como objetos media, contextos y switch.

El elemento <*importedDocumentBase*> permite importa documentos NCL. Esta compuesto por la etiqueta <*importNCL/*>, que posee un **alias,** a través del cual se referenciará tal documento, y **documentURI,** donde se especifica la ruta de ubicación del documento.

El elemento <*ruleBase*> permite la definición de reglas usadas en una aplicación NCL. Cada regla posee un **id**, una referencia a una propiedad de un objeto **(var),** un operador de comparación (**comparator**) y un valor(**value**). El valor de un atributo var debe ser una propiedad de un elemento media (de type="aplication/x-ncl-setting"). El comparator puede asumir cualquiera de los siguientes valores: **eq** (igual a...), **ne** (diferente de ...), **gt** (mayor que ...), **lt** (menor que ...), **gte** (mayor o igual que ...) y **lte** (menor o igual que ...). NCL también permite definir reglas compuestas a través del elemento <*composireRule*>. Las reglas son utilizadas para adaptación de contenido, a través del elemento <switch> y para adaptación de presentación, a través del elemento <descriptorSwitch>.

El elemento <*transitionBase*> permite la definición transiciones (posibilita pequeños efectos). Las transiciones se especifican a través del elemento <*transition*>. Las transiciones deben estar asociadas a un descriptor. Posee dos

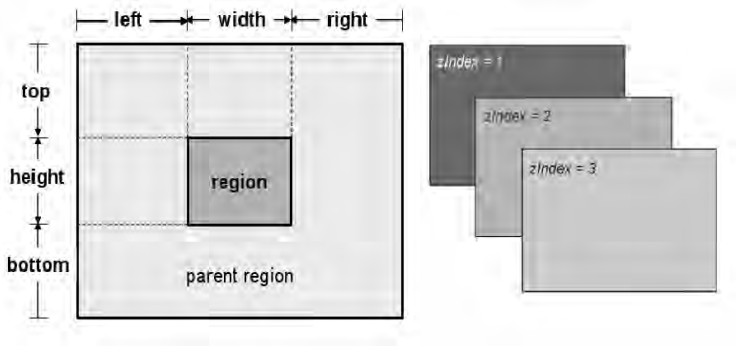


Figura 15: Parámetros del elemento <region>

atributos: **transIn** (identificador de transición que estable el inicio de la presentación) y **transOut** (identificador de transición que estable el finals de la presentación).

El elemento <*regionBase*> especifica los distintos espacios de exhibición de los media a través del elemento <*region*>. Todo elemento <*region*> posee un **id**, y atributos **(rigth, bottom, left, top, width, heigth, zIndex)** que definen el "área" de exhibición. Left, rigth, bottom top, width y heigth permiten especificar las dimensiones que tendrá la región (pueden expresarse en porcentaje ó en pixeles), zIndex permite identificar que media debe exhibirse cuando se tiene una sobreposición de medias (se exhibe aquel media que tenga asociada en su region mayor valor en el atributo zIndex).

Es posible importar regiones a través del elemento <*importBase*>, estas regiones deben estar situadas dentro de una región base de otro documento NCL. Este elemento posee los siguientes atributos: **alias**, nombre representativo que se utilizará para referenciar al documento; **documentURI**, para localizar el documento; **region**, en el caso de que el archivo importado contenga una region base, define que region de la aplicación contendrá las regiones importadas.

El elemento <*descriptorBase*> contiene un conjunto de elementos <*descriptor*> que especifican como los objetos media serán exhibidos. De igual manera permite la importación de descriptores a través del elemento <importBase>.

Cada elemento <descriptor> tiene asociado un **id**, y un un atributo **region,** el cual es un identificador de alguna región que será asociada a dicho descriptor. Todo objeto media que utilize un descriptor será exibido inicialmente en esa region. Ese atributo sólo precisa ser especificado para objetos que poseen un contenido visual. El atributo **explicitDur** define la duración de un objeto media asociado a tal descriptor. El valor de este atributo puede ser expresado en segundos. Si este atributo no se especifica, tomo como valor default (infinita). Otros atributos son **freeze** (identifica lo que ocurre al final de los medias asociados al descriptor, por ejemplo, en un video freeze="true" indica que al finalizar el video el último cuadro debe ser congelado indefinidamente) y **player** (identifica la herramienta de presentación a ser utilizada para exhibir medias asociados al descriptor)**.**

Dentro de un descriptor pueden ser especificados atributos para navegación, tales como **focusIndex, focusBorderColor, focusBorderWidth, focus-**

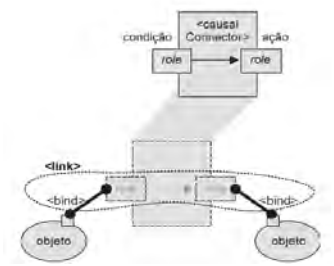


Figura 16: Ilustración de un link

BorderTransparency, defaultFocusBorderTransparency, focusSrc, focusSelSrc, selBorderColor, moveLeft, moveRight, moveUp, moveDown.

El elemento <*connectorBase*> contiene un conjunto de definiciones de distintos conectores, los cuales se especifican a través del elemento <*causalConnector*>.Los conectores definen el sincronismo, y en particular, la interactividad entre los objetos de una aplicación. Un elemento <*link*>asocia objetos a través de un conector, que define la semántica de asociación. Los link son definidos en el body y los conectores en el head (connectorBase). El elemento connectorBase permite la importación de conectores, al igual que los elementos regionBase y descriptorBase.

Un <causalConnector> representa una relación causal que puede ser utilizada por los <link>. En una relación causal, una condición debe ser satisfecha para que puedan ser disparadas ciertas acciones. Un causalConnector especifica estas relaciones a través de roles (**role**).

Un <link>, Figura 16, referencia a un causalConnector y define un conjuntos de mapeos. Posee un **id**, y un identificador de conector asociado a él, llamado **xconnector**. Otros elementos son: **<bind>,<linkParam>.** El elemento <bind> indica una ligadura entre una interface de componente y su rol. El elemento <linkParam> define un parámetro del link como un par [propiedad,valor]. Las propiedades y sus respectivos valores depende de la definición del conector al que esta asociado. Un link puede contener diversos bind y linkParam.

A su vez, bind posee un elemento <*bindParam*> que permite la definición de parámetros.

Los elementos de un <causalConnector> son: <*connectorParam*>, <*simpleCondition*>, <*compoundCondition*>, <*simpleAction*> y <*compoundAction*>.

ConnectorParam define parámetros de conector cuyos valores deberan ser definidos a través de links que utilizan el conector. SimpleCondition y compoundCondition definen las condiciones simples o compuestas de activación de los links que utilizan el conector. SimpleAction y compoundAction definen las acciones simples y compuestas que son realizadas cuando el link que utiliza el conector es activado.

Valores posibles para condiciones:

onBegin: cuando la presentación de un objeto ligado a ese rol es iniciada

...

onEnd: cuando la presentación de un objeto ligado a ese rol es finalizada

...

onAbort: cuando la presentación de un objeto ligado a ese rol es abortada ...

onPause: cuando la presentación de un objeto ligado a ese rol es pausada ...

onResume: cuando la presentación de un objeto ligado a ese rol es retomada luego de una pausa ...

onSelection: cuando una tecla es precionada en cuanto a un objeto ligado a ese rol ...

onBeginAttribution: antes de que un valor sea atribuído a una propiedad de objeto ligado a ese rol ...

onEndAttribution: después de que un valor ha sido atribuído a una propiedad de objeto ligado a ese rol ...

Valores posibles para acciones:

start: ... inicia la presentación del objeto asociado a ese rol

stop: ... termina la presentación del objeto asociado a ese rol

abort: ... aborta la presentación del objeto asociado a ese rol

pause: ... pausa la presentación del objeto asociado a ese rol

resume: ... retoma la presentación del objeto asociado a ese rol (si estaba en pausa)

set: ... establece un valor a la propiedad asociada a ese rol

Las relaciones definidas en un causalConnector son basadas en eventos. Un evento es una ocurrencia en el tiempo que puede ser instantánea o tener duración. En NCL se pueden distinguir 3 tipos de eventos:

Evento de presentación: presentación de un subconjunto de unidades de información de un objeto media.

Evento de selección: selección de un subconjunto de unidades de información de un objeto media.

Evento de composición: presentación de estructura de un nodo de composición. Son utilizados para representar el mapa de composición.

Los elementos <*meta*> y <*metadata*>permiten representar metadatos. El <meta> permite representar metadatos simples, con una propiedad (**name**) y un valor o lista de valores (**content**). El elemento <metadata> es destinado a la representación de metadatos más estructurados. Un metadato no contiene información de contenido utilizado para exhibir durante una presentación de un documento, sino que contiene información sobre el contenido utilizado o exhibido que puede ser procesada automáticamente.

El elemento <*port*> es un punto de interface de un contexto que ofrece acceso externo al contenido del contexo. Indican por donde comienza la exhibición. Posee un **id**, y un **component,** que es el objeto de media o contexto se exhibira.

Los elementos <*media*> tiene un identificador (**id)**, para que pueda ser referenciado porteriormente. También posee un atributo denominado **src**, que contiene un URI para su localización, ademas del atributo **descriptor**, que referencia a un elemento del head que contendrá información sobre el objeto media. Otro atributo es **type** (opcional) el cual define el tipo de media.

Los elementos <*area*> delimitan trechos tanto en tiempo como en espacio. Este elemento posee tres atributos, un **id, begin, end**, estos últimos dos permiten establecer intervalos de tiempos. También se puede definir un objeto media con el tag <*property*>, el cual define una interface para poder cambiar el valor de la propiedad en elobjeto media. Este tag posee un nombre (**name**) y un valor el cual tomará la propiedad **(value).**

Para objetos media de audio se identifica el atributo **soundLevel**, para medias visuales **location, bounds**, y para medias de texto se identifica **fontColor, fontSize, weigth y fontFamily**.

El elemento <*context*> agrupa objetos y links. Un contexto puede anidar otros contextos o switchs, salvo contextos recursivos. Permiten el encapsulamiento y modularidad. Posee un **id,** y un **refer**, que referencia a otro contexto previamente definido.

Un elemento <switch> es una composición que contiene varias alternativas. La decisión sobre cual será seleccionado es dada por reglas, definidas a través de <*bindRule*>. Así mismo se puede definir un opción por default dada por <*defaultComponent*>. Cabe destacar que todos los identificadores (id) deben ser únicos.

En NCL la realización de algunas tareas es complicada sin auxilio imperativo, tal como el procesamiento matemático, la manipulación sobre textos, animaciones complejas, etc, en fin, de modo general, tareas que necesiten de la especificación de algoritmos y estructuras de datos que son provistos de forma nativa por el lenguaje imperativo.

Un objeto media con código imperativo es definido en NCL por el elemento <media> con el atributo **type** reciviendo el valor "application/x-???", donde ??? depende del lenguaje imperativo usado. Por ejemplo, "application/x-ncl-NCLua" es usado para objetos con código Lua. El atributo **src**, debe referenciar a la localización del código imperativo que compone el contenido del objeto. Así, el ciclo de vida de un objeto media imperativo es controlado por el formateador NCL. Éste es el responsable de disparar la ejecución del objeto y mediar la comunicación de ese objeto con otros objetos del documento NCL.

Una vez instanciado el objeto, el exhibidor de objetos imperativo ejecuta un procedimiento de inicialización, el cual debe ser especificado. Este procedimiento es ejecutado una sola vez, para cada instancia, y crea todos los trechos de código y estructura de datos que pueden ser usados durante la ejecución de objetos. Ese procedimiento es también el responsable de registrar uno o más "tratadores de eventos" para la comunición con el formateador NCL. Después de este procedimiento de inicialización, la ejecución del objeto imperativo se torna orientado a eventos, es decir, cualquier acción comandada por el formateador NCL es entregada a los tratadores de eventos registrados, y cualquier modificación en la máquina de eventos, comandadas por instrucciones de código imperativo, son enviadas como eventos al formateador NCL.

C. El Lenguaje Lua

En la Figura 17 se exhibe en el centro un NCLua, rodeado por varias entidades con las que puede interactuar. Para comunicarse con un NCLua, una entidad externa debe se insertada en una cola de eventos, que luego se redirige a las funciones de control de eventos, definido por el programador de la secuencia de comandos NCLua. Los eventos son tratados de manera individual, cuando hay un evento en tratamiento, el resto de eventos permanece en la cola.

La principal ventaja del uso del paradigma orientado a eventos es que permite modelar los distintos acoplamientos entre entidades del sistema.

Para ser informado cuando eventos externos son recibidos, un NCLua debe registrar por lo menos una función de tratamiento en su cuerpo a través de una

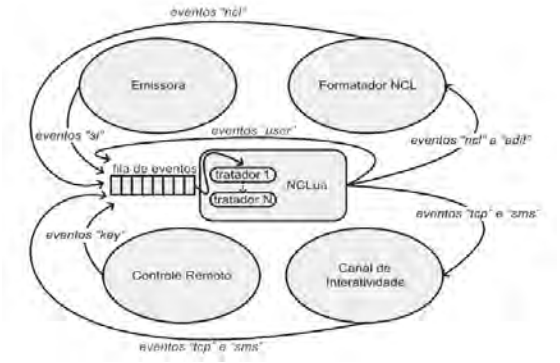


Figura 17: Paradigma de programación orientado a eventos

llamada a la función *event.registe.* El código de un NCLua por lo general sigue la siguiente estructura:

... - código de inicialización

function tratador(evt) - handler

... - código de un tratador

end

event.register(tratador) - registro de por lo menos un tratador

El código de incialización, la definición del tratador y su registro son ejecutados antes que el documento NCL señalize cualquier evento a NCLua, inclusive el de inicio de presentación del objeto. Después de este proceso de carga del script, efectuado por el sistema, el código del tratador es llamado cada vez que ocurre un evento externo. El código de inicialización puede ser utilizado para crear objetos y funciones auxiliares que serán usadas por los tratadores.

Los eventos son representados por tablas Lua con claves y valores que describen sus atributos. Por ejemplo, la función tratadora puede recibir un evento (parámetro *evt* del tratador) indicando que la tecla roja del control remoto fue presionada por el teleespectador. Este evento se representa como:

evt = { class = 'key', type = 'press', key = 'RED' }

La función *event.post* es utilizada para que un NCLua pueda por ejemplo, enviar datos por el canal de interactividad o señalizar su estado a un documento NCL. Como un NCLua debe ejecutarse rápidamente, la función de envío de eventos nunca aguarda el retorno de un valor. En caso de que el destino necesite retornar una información a NCLua, el lo hará a través del envío de un nuevo evento.

C.0.1. Clases de eventos

El campo *class* de una tabla que representa un evento es obligatorio y tiene como finalidad separar los eventos en categorías. La clase identifica no sólo el

origen de los eventos pasados a los controladores, sino también su destino, si el evento se genera y envia por un guión NCLua.

Las clases de eventos se definen los siguientes:

• **Clase ncl:** se utiliza en la comunicación entre un NCLua y NCL que contiene el objeto multimedia.

• **Clase key:** representa el presionamientode teclas del control remoto del usuario.

• **Clase tcp**: permite el acceso al canal interactivo a través del protocolo tcp.

• **Clase sms**: se utiliza para enviar y recibir mensajes SMS en dispositivos móviles.

• **Clase edit**: permite que los comandos de edición en vivo sean disparados a partir de scripts NCLua.

• **Clase si**: proporciona acceso a una gama de información multiplexada en un flujo de transporte y se transmitas periódicamente por difusión.

• **Clase user**: a través de esta clase, las aplicaciones pueden ampliar sus funcionalidades mediante la creación de sus propios eventos.

Dados los tipos de eventos NCL soportados, el campo *type* de un evento de clase ncl puede asumir dos valores "*presentation*" o "*attribution*". Los eventos de tipo "*selection*" son tratados por la clase key.

Los eventos de presentación están asociados a la presentación de contenido, que son identificados por el campo *l*abel del evento. El campo *a*ction indica la acción a ser realizada o señalizada por NCLua, dependiendo de ésta, NCLua puede estar recibiendo o generando el evento.

Un evento de presentación posee la siguiente estructura:

• *class:* 'ncl'

• *type:* 'presentation'

• *label:* [string]

• *action:* [string]- - puede asumir los valores 'start', 'stop', 'abort', 'pause', o 'resume'.

Los eventos de atribución están asociados a las propiedades del objeto NCLua, que son identificadas por el campo *name*. El campo value posee el valor a ser atribuído a la propiedad y es siempre una cadena. La acción 'start' en un evento de atribución corresponde al rol 'set' de un link NCL.

Un evento de atribución posee la siguiente estructura:

• *class:* 'ncl'

• *type:* 'attribution'

• *name:* [string] - - nombre de la propiedad asociada al evento.

• *action:* [string] - - puede asumir los valores 'start', 'stop', 'abort', 'pause', o 'resume'.

• *value:* [string] - - nuevo valor a ser atribuído a la propiedad.

D. Breve Descripción del Diseño de la Aplicación

El diseño de la aplicación fue desarrollado siguiendo la metodología proporcionada por el lenguaje conceptual NCM. Cada objeto media se asocia a una región a través de un descriptor. Por lo cual cada objeto a visualizar en la pantalla tiene asociado su correspondiente region y descriptor. Como consecuencia de las dimensiones y complejidad en el modelado de toda la aplicación, se han

adoptado algunas convenciones en el diseño. En la Figura 18 se muestra como se disea un conector, el cual asocia una acción a una condición.

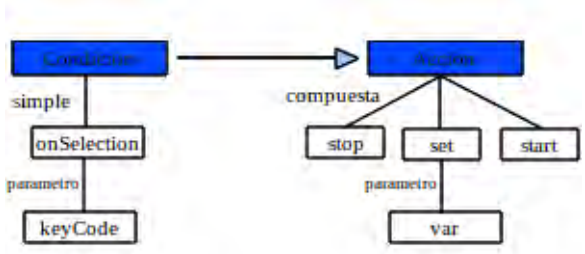


Figura 18: Conector onKeySeletionStopSetStart

En la Figura 19 s-e utiliza la simbología descripta en el libro "PROGRAMANDO EM NCL" la cual fue explicada anteriormente en la sección 3 (Nested Context Languaje (NCL))

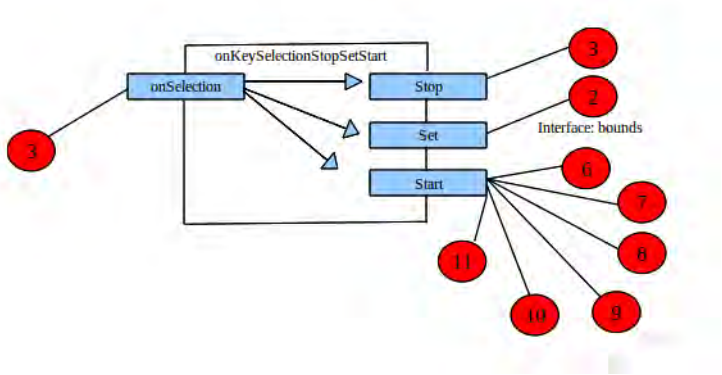


Figura 19: Ejemplo diseño de un link

En la Figura 20 se muestra el diseño completo de un prototipo para la aplicación.

D.1. Limitaciones y dificultades observadas

Durante el desarrollo de la aplicación se observaron algunos conflictos, entre los cuales se pueden mencionar:

• El uso de muchos media de textos parece ser lo más lógico, dado a que son formatos más livianos en comparación con imagenes, pero resulta ser que la randarización de imagenes es mucho más eficiente que la de texto.
• En una primera instancia el menú principal de la aplicación se pensó usando focus (atributo de descriptor), es decir, declarando para cada opción un media texto. Al desplazarnos por las opciones se observó que el foco quedaba

en los objetos que se atravesaban anteriormente, lo que se debe a un error en la implementación de la versión de GINGA utilizada.

• En la implementación de los test en primera instancia se realizó completamente en NCL, pero observamos que se necesitaban muchas líneas de código, e incluso era imposible no repetir código dado a que no es posible modularizar.

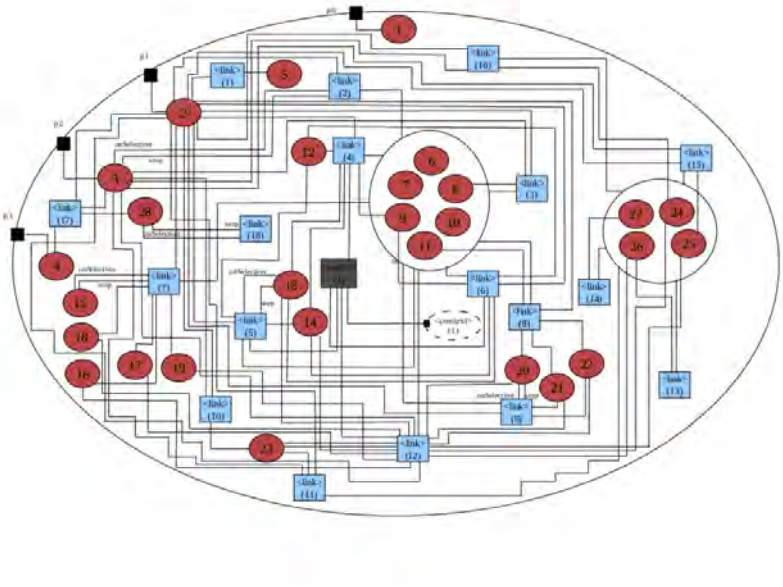


Figura 20: Diseño de un prototipo