# CreaTVDigital: Composición Visual de Aplicaciones Interactivas para TV Digital

Marcelo Arroyo[*], Sebastián Schwartz[**], Sergio Cardozo[**] y Laura Tardivo[*]

**Resumen**

En este artículo se describe el desarrollo y uso de *Crea TV Digital*, una aplicación libre para el desarrollo visual de aplicaciones de televisión digital interactiva. La herramienta está orientada a productores de contenidos audiovisuales que no tengan conocimientos de programación de este tipo de aplicaciones.

Las aplicaciones interactivas en el sistema de televisión digital argentino se basan en el middleware Ginga. En particular ésta herramienta está basada en la especificación Ginga-NCL.

Si bien existen otras herramientas que asisten al desarrollo de aplicaciones Ginga, éstas están mas orientadas a desarrolladores de software, las cuales no son utilizables directamente por los profesionales de producción o edición de contenidos audiovisuales, por lo cual ésta aplicación permite acercar el desarrollo de programas Ginga-NCL a profesionales no programadores. Además permite el desarrollo rápido de prototipos, correctos por construcción, en un ambiente de desarrollo integrado con ginga-ar para la prueba de aplicaciones.

## 1. Introducción

Desde la introducción del sistema de televisión digital (TVD) en la República Argentina en 2009, se iniciaron varios proyectos de investigación y desarrollo alrededor de esta tecnología.

En la Universidad Nacional de Río Cuarto, tratando de contribuir al desarrollo del sistema y en particular en el Departamento de Computación de la Facultad de Ciencias Exactas Físico-Químicas y Naturales, se dieron inicio a dos proyectos de desarrollo sobre aplicaciones relacionadas con la televisión digital interactiva.

Uno de esos proyectos fue el desarrollo de una aplicación concreta sobre un video educativo producido por la Facultad de Agronomía y Veterinaria sobre la enseñanza de suturas, denominado *Punto x Punto suturas*[3]. La interactividad permite a los alumnos de veterinaria poder acceder a información adicional sobre técnicas e instrumental, gráficos detallados y también, el desarrollo opcional de una pequeña autoevaluación. Ese primer trabajo fue desarrollado en el marco de un trabajo final de carrera de Analista en Computación[5].

Esta primera experiencia llevó a la detección que no existía ninguna herramienta orientada a técnicos y profesionales de producción audiovisual, lo que generó el desarrollo de *Crea TV Digital*, un proyecto de tesina de la carrera de Licenciatura en Ciencias de la Computación. Los objetivos del proyecto fue el desarrollo de una herramienta de composición visual de aplicaciones de TVD interactiva sin necesidad de contar con conocimientos de programación NCL-Lua[6] y que genere código Ginga-NCL.

En este trabajo se presentan las características del producto obtenido, sus objetivos y conceptos utilizados, detalles de diseño e implementación y algunos ejemplos de su uso y aplicación.

*Crea TV Digital* es un paquete de software libre, multiplataforma, el cual puede ser descargado desde `http://code.google.com/p/creatvdigital/`.

[*]Departamento de Computación - FCEFQyN - Universidad Nacional de Río Cuarto.
[**]NeoSur S.A. - Consorcio de Televisión Digital.

En la sección 2 se da una breve introducción al sistema de televisión digital argentino. En la sección 3 se da una descripción del middleware *Ginga* y los lenguajes de desarrollo de aplicaciones NCL y Lua. En la sección 4 se describen los conceptos básicos utilizados en *Crea TV digital* para la creación y edición de una aplicación interactiva y algunos ejemplos de su utilización. En la sección 5 se analiza su diseño y algunos detalles de implementación. Finalmente, se analizan algunas extensiones como trabajo futuro.

En el anexo A se muestra el desarrollo paso a paso de una aplicación simple como ejemplo. En el anexo B se describe el proceso de compilación e instalación de *Crea TV Digital*.

# 2. Televisión Digital en Argentina

Un sistema de televisión digital provee muchas ventajas con respecto a los sistemas de televisión analógica. Los avances tecnológicos, tanto en los sistemas de telecomunicaciones, como también los avances logrados con los algoritmos de compresión de datos, permiten que la digitalización de una señal emitida pueda aprovechar al máximo el ancho de banda de los enlaces de comunicación (permitiendo transmitir simultáneamente varias señales de video y audio), como así también la calidad de servicio, como por ejemplo, la ausencia de *fantasmas* y *nieve* gracias a los algoritmos de detección y corrección de errores.

La transmisión y recepción de señales digitales de televisión permiten no sólo aumentar la calidad de imagen y sonido (alta definición y fidelidad), sino que también permiten incluir en la misma señal, además del video y audio principal, la transmisión de otros objetos de datos como programas de computadora y otros objetos multimedia (imágenes, sonidos, videos, texto, HTML, etc).

Un sistema de televisión digital se ejemplifica en la figura 1.

El sistema de recepción comprende un televisor, un receptor, el cual es un sistema de computación conteniendo componentes de hardware especializado para el manejo (decodificación, descompresión, detección y corrección de errores, etc) de las señales recibidas y un sistema de software que será el responsable de ejecutar las aplicaciones recibidas o residentes en el sistema.

Actualmente hay disponibles en el mercado equipos de televisión con todos estos componentes integrados.

Como dispositivo de interacción con el equipo receptor se dispone de un control remoto, el cual también está estandarizado.

La República Argentina adoptó en 2009 las normas SBTVD (Sistema Brasileño de Televisión Digital). La norma ISDB-T especifica la codificación de datos y las especificaciones de transmisión para radiodifusión de televisión digital (TVD) terrestre[1].

Este conjunto de normas también establecen las características de la plataforma de software que deberá contener cualquier receptor y los lenguajes y ambientes de programación de las aplicaciones.

En particular, la República Argentina, adoptó como obligatorio la implementación del middleware con soporte a aplicaciones NCL-Lua (conocida como Ginga-NCL)[1]. En Brasil se desarrolló un prototipo de implementación de referencia, conocido como *Ginga*. Ginga fue liberado bajo la licencia GNU y fue tomado por un laboratorio de investigación y desarrollo argentino, LIFIA[1], donde se realizaron importantes contribuciones y mejoras a la implementación original. El resultado de este trabajo se conoce actualmente como la versión *ginga-ar*[9].

El sistema de televisión digital permite la transmisión (broadcast) de datos, además del video y audio principal, durante la emisión de una programación[2], tal como se muestra en la figura 1.

Estos datos se almacenan en el receptor[3] en un sistema de archivos. Como ya hemos descripto, estos archivos pueden contener datos de diferentes tipos y formatos como imágenes, sonidos, videos, contenido HTML, texto plano y en particular, programas de computación. Estos programas o aplicaciones, una

[1] *Laboratorio de Investigación y Formación en Informática Avanzada*. Facultad de Informática, Universidad Nacional de La Plata. Argentina.

[2] Esto se conoce como un programa no lineal.

[3] Comúnmente conocido como *decodificador* o *setopbox*.

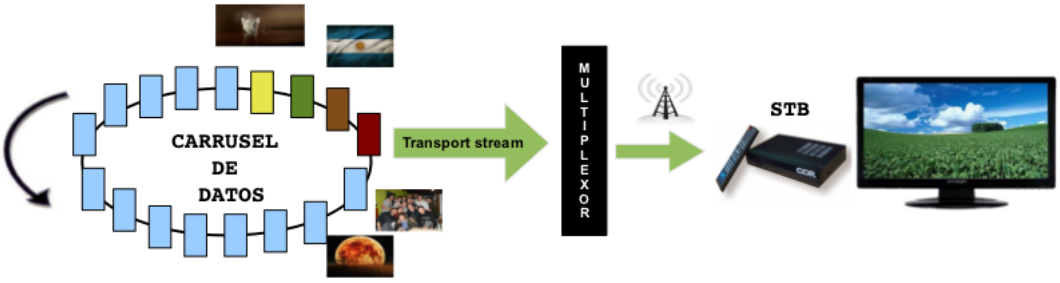


Figura 1: Sistema de TV digital

vez que arribaron al receptor son ejecutados (interpretados) por el middleware GINGA el cual es el responsable de la ejecución de las aplicaciones.

Una aplicación de televisión digital interactiva ofrece posibilidades al televidente a realizar acciones usando su control remoto. El programa reacciona ante los eventos ocurridos y realiza alguna acción, como por ejemplo, mostrar alguna información extra sobre el video principal, redimensionamiento de éste, etc. Estos programas de TV digital se conocen como *programas no lineales*[2].

De esta forma, el receptor constituye un completo sistema de computación, en el cual se pueden ejecutar programas los cuales pueden o no estar asociados a la programación actualmente en emisión (video y sonido principal).

Este sistema permite que el telespectador pueda convertirse en un televidente activo, interactuando con el sistema y permitiendo visualizar diferentes componentes y alternativas ofrecidos por el programa.

Si se dispusiera de un sistema de comunicación desde los telespectadores hacia la planta de transmisión u otro destino, una aplicación podría enviar información suministrada por cada telespectador lo que abre inmensas posibilidades para el desarrollo de aplicaciones basados en televisión digital. Este canal de comunicación se denomina *canal de retorno*.

Aún sin el canal de retorno, el telespectador tiene acceso a una nueva forma de ver televisión, interactuando localmente con el *setopbox* el cual le permitirá el acceso a elementos multimedia adicionales y hasta podrá realizar operaciones (generalmente opcionales) como por ejemplo, acceder a información extra dependiente del contexto actual.

Los televidentes que acceden a servicios de TVD están acostumbrados a interactuar con algunas de estas aplicaciones como la guía de programación o en programas especialmente creados con interactividad, como por ejemplo, partidos de fútbol con acceso a repeticiones de los goles, acceso a la formación de los equipos, etc.

# 3. Middleware Ginga

El middleware *Ginga*[9] es una capa de software en el sistema receptor que corre por encima de un sistema operativo y es responsable de crear un ambiente de ejecución apropiado para las aplicaciones de TV digital. El middleware define, entre otras cosas, una interface para el programador de aplicaciones.

El middleware Ginga ofrece mecanismos para la sincronización espacial y temporal de objetos multimedia y provee dos ambientes de ejecución de aplicaciones: un ambiente declarativo denominado Ginga-NCL y un ambiente imperativo llamado Ginga-J.

El primero está basado en un lenguaje de programación declarativo basado en XML, denominado *Nested Context Language (NCL)*, el cual permite la definición de elementos multimedia, características de su reproducción y modos de interacción en una manera declarativa[4].

El segundo ambiente, Ginga-J, está basado en el lenguaje de programación Java.

[4]Un lenguaje declarativo permite especificar el *qué*, mientras que uno imperativo especifica el *cómo*.

En éste trabajo se describe el ambiente declarativo Ginga-NCL, en el cual está basado el desarrollo de la versión actual de *Crea TV Digital*.

## 3.1. Modelo de contextos anidados

Todo lenguaje declarativo del tipo XML se basa en algún modelo conceptual de datos o *Nested Context Model* (NCM), el cual a su vez, determina los conceptos estructurales de los datos, sus relaciones y acciones a ejecutar (respuestas) ante la ocurrencia de eventos. La figura 2 muestra el modelo conceptual básico NCM, el cual describe los posibles tipos de objetos.

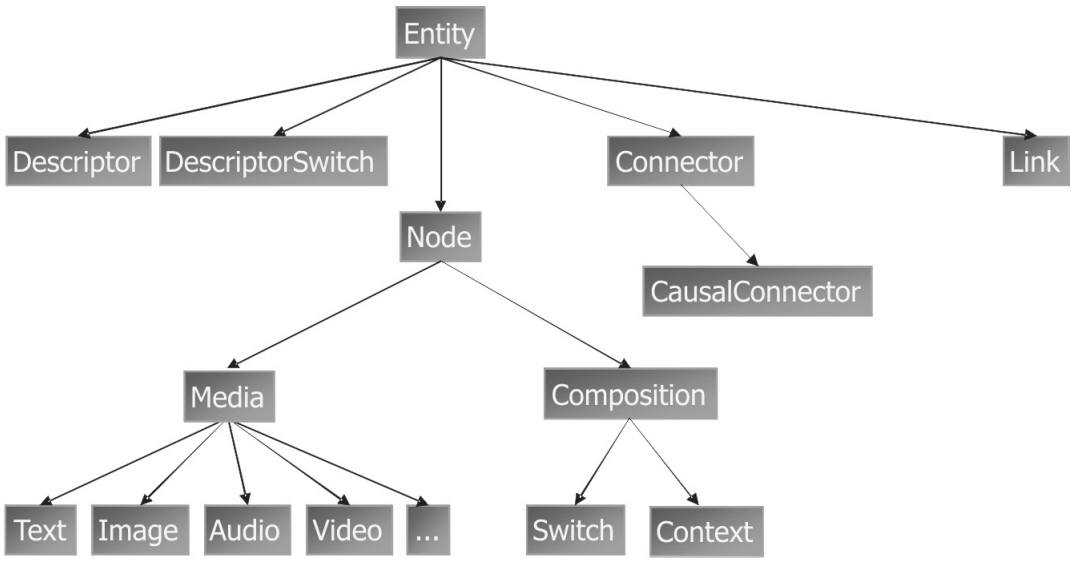


Figura 2: Nested Context Language

Un *nodo* es la descripción de un objeto de contenido multimedia, el cual puede ser simple o compuesto. Un nodo contiene *propiedades* y *anclajes*[5] y definen la *interface* del objeto.

Un *descriptor* describe la forma en que un nodo puede ser exhibido, por ejemplo en qué región de la pantalla y su duración de reproducción.

Un *conector* define una especie de esquema o patrón de relación entre objetos y eventos. Una instancia concreta de una relación se realiza por medio de un *link*, el cual asocia objetos y eventos concretos en un conector.

Un nodo de *contexto* agrupa objetos (nodos, descriptores, conectores, enlaces, etc) y representa una especie de subsistema de una aplicación.

Con cada contexto se pueden asociar puntos de acceso al mismo, denominados *ports*. Un *port* representa un punto de ingreso a un contexto.

La definición de descriptores y contextos permite formar especies de *bibliotecas* de código NCL o componentes de software reusables.

## 3.2. Nested Context Language

El *Nested Context Language* (NCL) se basa en el NCM descripto en la sección anterior.

Es un lenguaje basado en *e**X**tensible **M**arkup **L**anguage* (XML) y define elementos (*tags*), propiedades y características estructurales para cada objeto definido en el NCM.

La figura 3 muestra la estructura general de un programa NCL.

La sección `head` incluye las declaraciones de regiones espaciales para los objetos multimedia, sus propiedades de reproducción (descriptores) y los patrones de la interactividad provista (conectores).

[5]Un *anclaje* es un enlace o referencia al repositorio de un objeto.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<ncl id="main" xmlns="http://www.ncl.org.br/NCL3.0/EDTVProfile">
<head>
  <!-- regions, descriptors, connectors, ... -->
</head>
<body>
  <!-- medias, ports, switchs, links, ... -->
</body>
</ncl>
```

Figura 3: Estructura de un programa NCL

En el cuerpo (`body`) del documento se incluyen las descripciones de los objetos multimedia, aspectos temporales de reproducción y las acciones concretas generadas por la ocurrencia de eventos.

Hay dos tipos de eventos: aquellos generados por el pasaje de tiempo (inicio y final de reproducción de objectos multimedia) y aquellos eventos generados producto de acciones del televidente (por el pulsado de botones del control remoto).

Las regiones espaciales se definen en contenedores de regiones como se muestra en el siguiente framento de programa de ejemplo, en los cuales se definen atributos como posición, tamaño y plano o capa (`zIndex`):

```
<regionBase>
  <region id="sr" left="0" top="0" height="100%" width="100%" zIndex="0"/>
  <region id="br" left="620" top="478" height="100" width="100" zIndex="1"/>
  <region id="vr" left="75%" top="0" height="25%" width="25%" zIndex="1"/>
</regionBase>
```

Cada region debe tener sus propiedades o descriptores, indicando a cada objeto multimedia qué región le corresponde y otros atributos como por ejemplo, la duración de exhibición. Los descriptores se agrupan en *bases o conjuntos de descriptores*.

```
<descriptorBase>
  <descriptor id="sd" region="sr"/>
  <descriptor id="bd" explicitDur="42.8s" region="br">
  <descriptor id="vd" region="vr"/>
</descriptorBase>
```

Un conector es un patrón[6] de interactividad. La idea de que los conectores definen patrones abstractos los cuales luego pueden instanciarse con objetos concretos permite la definición de patrones de interactividad reusables.

A modo de ejemplo el siguiente conector relaciona dos objetos (abstractos) y define el comportamiento que un objeto multimedia debe comenzar a reproducirse (rol *start*) cuando comience la exhibición de otro (rol *onBegin*):

```
<causalConnector id=    onBeginStart    >
  <simpleCondition role=   onBegin     />
  <simpleAction role=    start     />
</causalConnector>
```

Es posible cargar conectores ya definidos en otros archivos NCL, como en el siguiente ejemplo:

```
<connectorBase>
  <importBase documentURI= "../connector/causalConnBase.ncl" alias="conEx"/>
</connectorBase>
```

En el `body`, el cual es el contexto principal, se deberá incluir al menos una puerta de ingreso al contexto, indicando qué objetos multimedia deberán reproducise inicialmente. Por ejemplo:

```
<port id="entry" component="mv"/>
```

Las puertas (*ports*) son puntos de entrada a contextos permitiendo especificar aplicaciones concretas en las cuales se relacionan estos componentes de software.

[6]Plantillas genéricas que luego pueden instanciarse con objectos concretos.

Aquí también se describen los objetos multimedia utilizados. Un objeto multimedia puede incluir aspectos de reproducción en el tiempo por medio del elemento `area`, el cual incluye los atributos *begin* y *end* permitiendo especificar tiempos de inicio y final de su reproducción (sin necesidad de interactividad por parte del usuario).

```
<media id="mv" src="main.mpg" type="video/mpeg" descriptor="sd"/>
<media id="button" src="button.png" descriptor="bd">
  <area id="ba" begin="0s" end="42.8s"/>
</media>
<media id="video" src="video.mpeg" type="video/mpeg" descriptor="vd"/>
```

Finalmente, también en el `body`, se deberán instanciar (con elementos `<link>`) los patrones de conexión (`connectors`) entre objetos y roles concretos. A continuación se muestran tres instanciaciones de conectores concretos para el manejo de eventos:

El primer `link` instancia un conector *onBeginStart* ligando el rol (evento) `onBegin` del objeto `mv` (video principal) para que también se inicie la exhibición del objeto `button` (imagen del botón).

El segundo `link` crea una instancia concreta del conector `onKeySelectionStart` vinculando el evento de pulsado del botón azul del control remoto con el inicio de la reproducción del objeto `video`.

El tercer `link` instancia el conector `onEndStop` para que al final de la reproducción de `button` también finalice la reproducción del elemento `video`:

```
<link id="beginbtn" xconnector="conEx#onBeginStart">
    <bind role="onBegin" component="mv"/>
    <bind role="start" component="button"/>
</link>
<link id="bluebtn" xconnector="conEx#onKeySelectionStart">
  <bind role="onSelection" component="button" interface="ba">
    <bindParam name="keyCode" value="BLUE"/>
  </bind>
  <bind role="start" component="video"></bind>
</link>
<link id="endvideo" xconnector="conEx#onEndStop">
  <bind role="onEnd" component="button"/>
  <bind role="stop" component="video"/>
</link>
```

El elemento NCL `bind` permite asignar roles (eventos) a un objeto multimedia. El elemento `bindParam` permite asociar a un rol con parámetros del rol. En el ejemplo de arriba se instancia el parámetro `keyCode` con el valor `BLUE` en el rol `onSelection`, lo cual corresponde con el pulsado del botón azul del control remoto.

Este es un ejemplo de un rol parametrizado.

Con *Crea TV Digtal* esta aplicación puede ser desarrollada de manera interactiva sin necesidad de conocer el lenguaje NCL.

El ambiente Ginga-NCL también permite el uso del lenguaje de programación `Lua`[8] integrando la aplicación NCL con invocación a funciones en algún script.

Generalmente las funciones Lua son invocadas al dispararse eventos por lo que esas funciones constituyen manejadores de eventos.

Lua es un lenguaje de scripting imperativo, orientado a objetos con el cual se pueden implementar algunas funciones como memorizado de datos (usando variables), con lo cual se pueden definir interacciones avanzadas y aplicaciones con estado, lo cual es prácticamente dificultoso lograr sólo con NCL.

En la versión actual, *Crea TV digital* no brinda ningún soporte para la generación de código Lua, pero es posible incluir scripts ya que NCL los trata como un tipo de objetos multimedia mas.

# 4. Crea TV Digital

El objetivo principal de *Crea TV digital* es lograr que una persona sin conocimientos de programación o el uso de lenguajes de la familia XML, pueda desarrollar una aplicación de TV digital interactiva de manera gráfica, intuitiva, rápida y sin errores.

Está destinada fundamentalmente a profesionales de producción audiovisual, por lo que una de sus principales características fue su interface de usuario debía ser familiar con las herramientas de edición de video y sonido con que están comúnmente familiarizados.

Hasta el momento, existen otras dos herramientas para asistir al desarrollo de aplicaciones NCL. La primera, la cual tiene prácticamente los mismos objetivos que *Crea TV digital*, se denomina *NCL Composer*[10], la cual existe desde hace varios años y recientemente se lanzó su nueva versión. La diferencia con *Crea TV digital* es que en NCL Composer, la vista temporal de exhibición de objetos y patrones de interacción está basada en el concepto de sistemas de transición de estados mientras que en *Crea TV digital* está basado en un modelo de líneas de tiempo.

La segunda herramienta, `NCL Eclipse`[11], que se puede descargar de `http://laws.deinf.ufma.br/~ncleclipse/`, es un *plug-in* del entorno integrado de desarrollo Eclipse[12] y está destinada a programadores, asistiendo en el proceso de escritura de aplicación, chequeando errores en tiempo de edición y otras características muy útiles para el desarrollador de aplicaciones NCL-Lua.

Esta última herramienta no es comparable con *Crea TV Digital* y *NCL Composer* ya que tienen diferentes objetivos y están destinados a diferentes tipos de usuarios.

*Crea TV digital*, de manera similar a *NCL Composer*, propone al usuario el desarrollo de una aplicación en tres pasos:

1. Seleccionar los elementos multimedia a utilizar en la aplicación.
2. Definir las regiones espaciales, definir sus propiedades y asociarlas a elementos multimedia.
3. Definir la interacción de los objetos multimedia en el tiempo y las reacciones ante la ocurrencia de eventos,

Ambas herramientas son muy similares conceptualmente para la realización de los primeros dos pasos. La diferencia fundamental está en el tercer paso, que se describe en la sección 6 ya que se basan en modelos visuales diferentes.

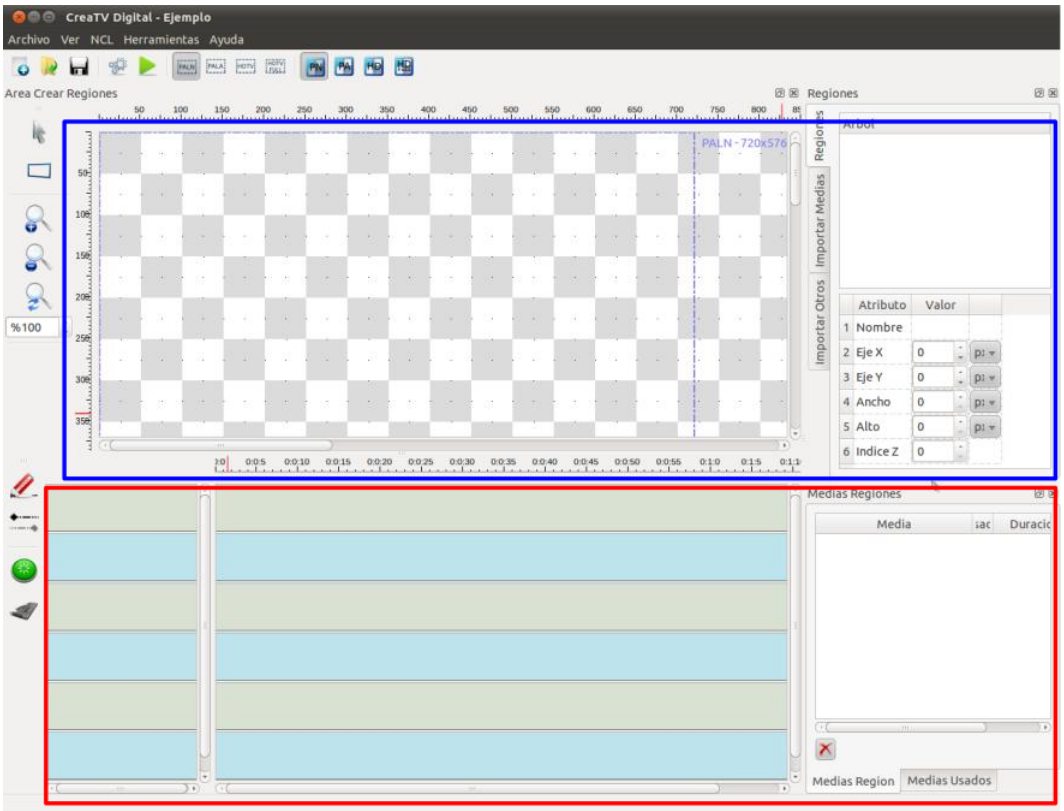


Figura 4: Interface de Crea TV digital

En la figura 4 se puede apreciar la interface del usuario de *Crea TV digital*. En la parte inferior derecha se listan los objetos multimedia utilizados en la aplicación y sus regiones asociadas. En la

parte superior (marcada en azul) está el área de definición de regiones con sus atributos o propiedades editables en la solapa de la derecha.

En la parte inferior (marcada en azul) se encuentra la sección de edición de interactividad basada en línea de tiempo.

Cada región definida en el área espacial se asociará a algún objeto multimedia, como se muestra en la figura 5.

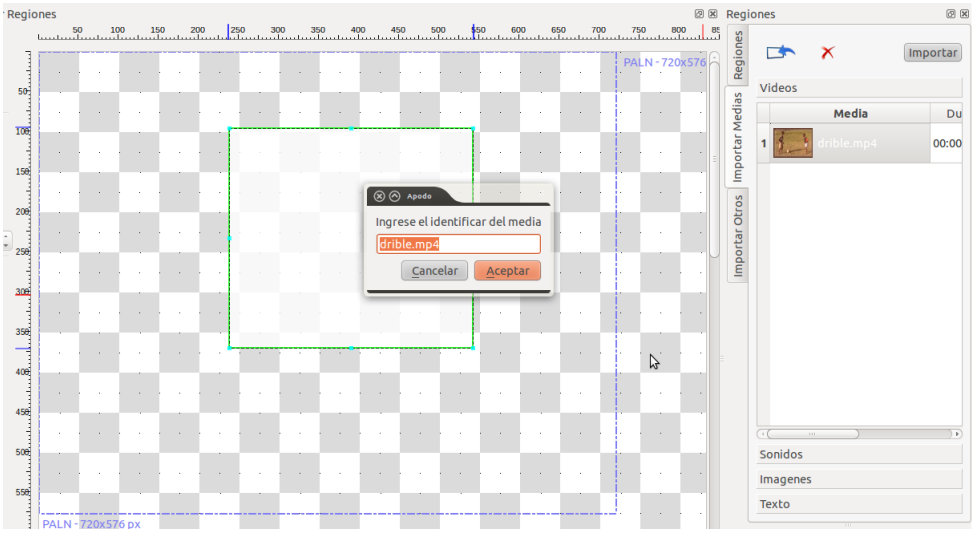


Figura 5: Asociación de un objeto multimedia con una región.

## 4.1. Interacción basada en línea de tiempo

El modelo de evolución temporal desarrollado se basa en el concepto de líneas de tiempo. Este concepto es muy utilizado en las herramientas de edición de video y sonido, por lo que los profesionales de desarrollo de producción audiovisual están muy familiarizados con este modelo.

En los sistemas de edición de video o sonido, conceptualmente, los únicos eventos a considerar están relacionados al pasaje del tiempo y eventualmente al comienzo y finalización de la reproducción de objetos de multimedia.

En una aplicación de TV digital, hay eventos adicionales para considerar: las reacciones de la aplicación ante la ocurrencia de eventos de interacción por parte del televidente con su control remoto.

La figura 6 muestra la forma de relacionar objetos multimedia en base a líneas de tiempo. En este caso se muestra que la imagen `boton.png` se comenzará a mostrar por el lapso de tiempo dado en la figura el cual es relativo al video principal.

Para habilitar la interactividad en un intervalo de tiempo determinado sobre algún objeto multimedia, se debe agregar un punto de interactividad, el cual se asociará a algún botón del control remoto y éste último se vinculará con otro objeto multimedia, tal como se puede apreciar en la figura 7. Este ejemplo significa lo siguiente:

1. El video principal (idMedia) es la puerta de entrada (del `<body>`).
2. La imagen *boton.png* se mostrará pasado un tiempo de reproducción.
3. Desde ese momento, si el usuario presiona el botón rojo del control remoto, se reproducirá el objeto *Imagen5*.

En el cuadro 1 se describe la semántica asociada a cada representación iconográfica de líneas de tiempo utilizadas.

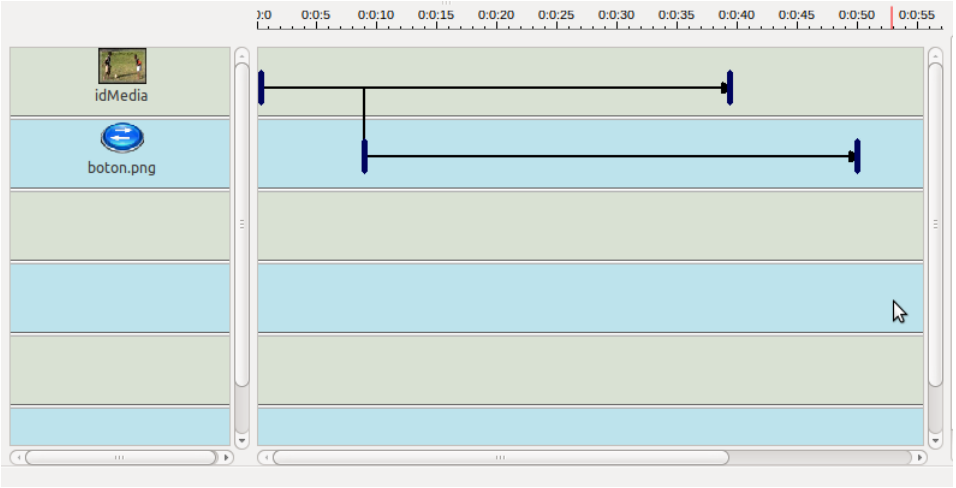


Figura 6: Creación de interacción sobre líneas de tiempo

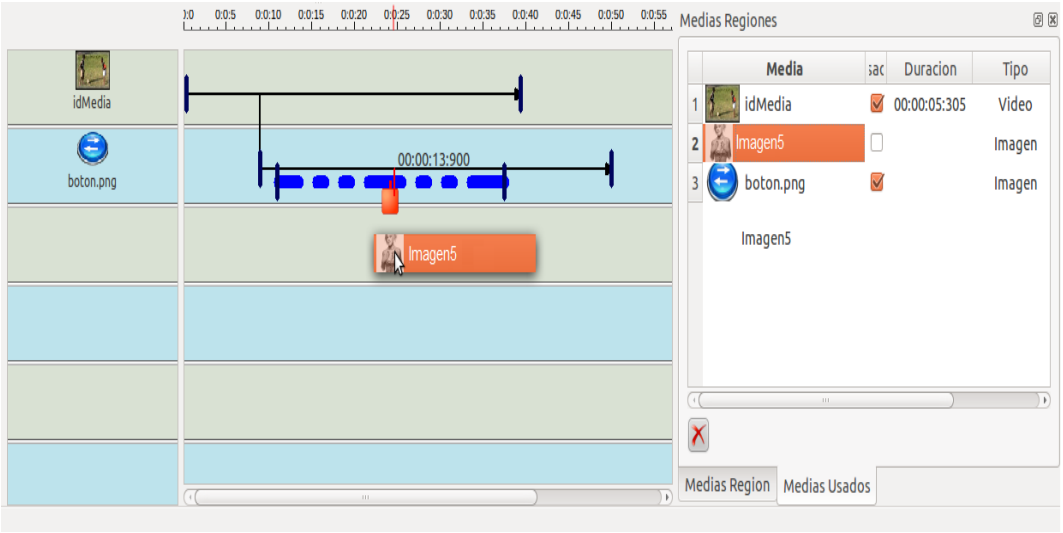


Figura 7: Creación de un intervalo de interactividad.

| Ícono | Descripción |
|---|---|
| ——————● | Finalización natural (sonido, video) |
| —[——]— | Inicio y fin de intervalo de tiempo |
| ——————♦(2,5) | Fin con tiempo explícito |
| ♦ – – – ♦ | Habilitación de interacción |
| ■ ■ ... 0 ... | Botones de interactividad |
| \| | (línea vertical) Conecta roles de interacción entre objectos. |

Cuadro 1: Semántica de los gráficos de líneas de tiempo.

Los intervalos de tiempo que habilitan interactividad se grafican con una línea punteada por debajo de la línea de tiempo de exhibición de un objeto multimedia y se conectan verticamente con una tecla o botón de interacción, el cual a su vez se conecta con el objeto multimedia a activar como respuesta de interactividad, tal como se muestra en la figura 8.

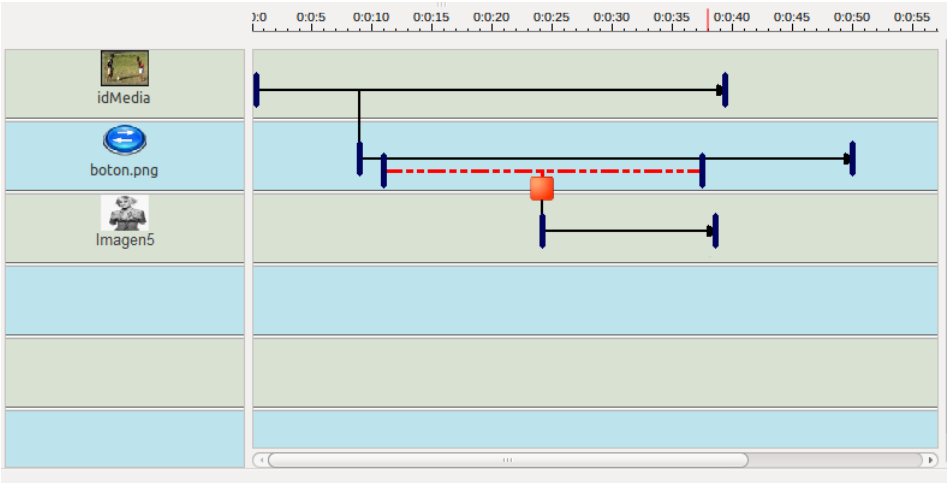


Figura 8: Línea de tiempo de interactividad.

La figura 9 muestra la visión estructural de una aplicación[7] , la cual es básicamente un grafo de transición de estados, donde los nodos pueden ser objetos multimedia, puertos y botones (teclas) de interacción y los arcos rotulados describen relaciones *evento/efecto*.

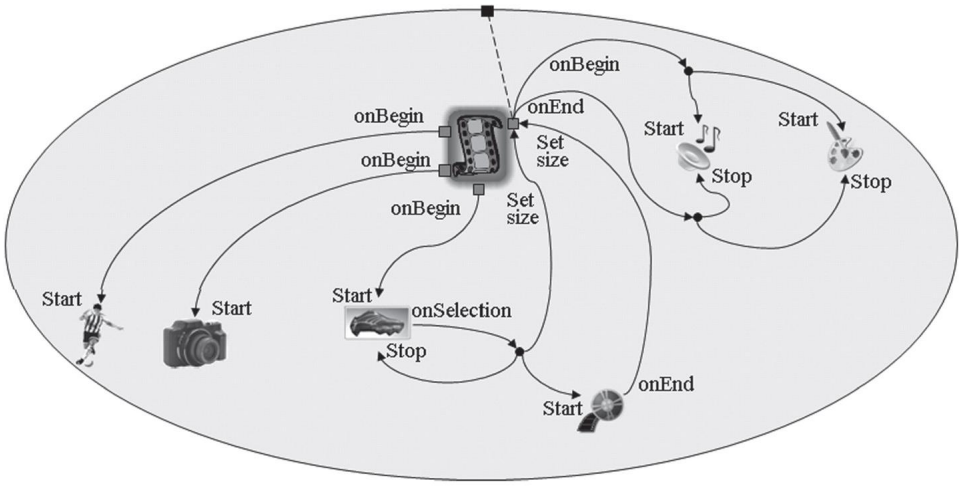


Figura 9: Visión estructural.

En aplicaciones complejas, con una gran cantidad de elementos multimedia y muchos puntos de interactividad, la visión estructural puede ser difícil de comprender y no provee un gráfico natural con respecto al pasaje del tiempo.

La figura 10 muestra la misma aplicación con el modelo basado en línea de tiempo. Este gráfico

[7]Ejemplo *O primero joâo*, extraído de [2].

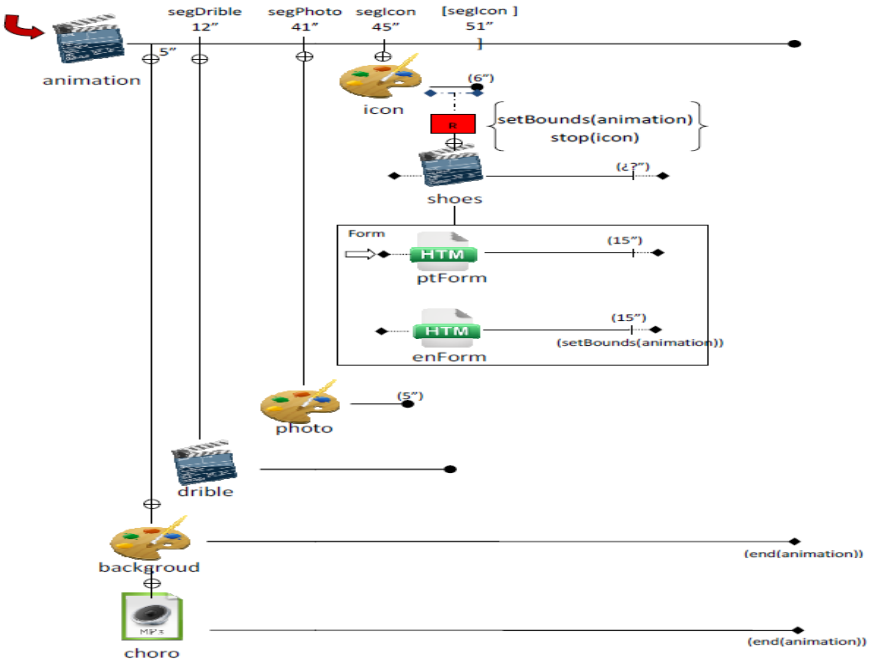


Figura 10: Vista basada en línea de tiempo.

presenta con mayor claridad la interacción entre los objetos multimedia y los posibles eventos de interacción a lo largo del pasaje del tiempo.

Las pruebas realizadas con desarrolladores de productos audiovisuales han mostrado que este modelo es mas intuitivo y familiar, mientras que el modelo estructural les ha costado mucho comprender, principalmente cuando el sistema de transición de estado contiene una gran cantidad de elementos multimedia (nodos) y eventos posibles.

# 5. Diseño e implementación

*Crea TV digital* fue diseñada usando técnicas de diseño orientadas a objetos[13]. El modelo de objetos de dominio (DOM) que representan las entidades de una aplicación forman un árbol de instancias de clases de objetos que representan cada uno de los elementos iconográficos utilizados. EL DOM tiene la estructura de la figura 11.

El diseño de la interface gráfica es una instancia del patrón *model-view-controller (MVC)* y está impuesto por la biblioteca de creación de interfaces gráficas utilizada.

La generación de código es un componente débilmente acoplado con el modelo de objetos de dominio y sigue el patrón *visitor*. Cada objeto de dominio es responsable por la generación de código de su propio nodo en el DOM. Esto facilita la implementación del generador de código el cual implementa un recorrido conveniente por el DOM invocando al generador de código NCL del nodo visitado.

Este diseño permite que puedan incluirse fácilmente otros generadores de código a otros lenguajes destino, como podría ser Ginga-J o para la generación de aplicaciones HTML5 con javascript.

El lenguaje de programación en que *Crea TV digital* fue implementada es C++ utilizando las bibliotecas Qt[14] (versión 4). Este framework permite el desarrollo de aplicaciones multiplataforma ya que está implementada en una gran variedad de sistemas operativos como GNU/Linux, MS-Windows, Mac OS-X y además está disponible para varias plataformas de dispositivos móviles como los teléfonos celulares de Nokia.

Las otras herramientas utilizadas son:

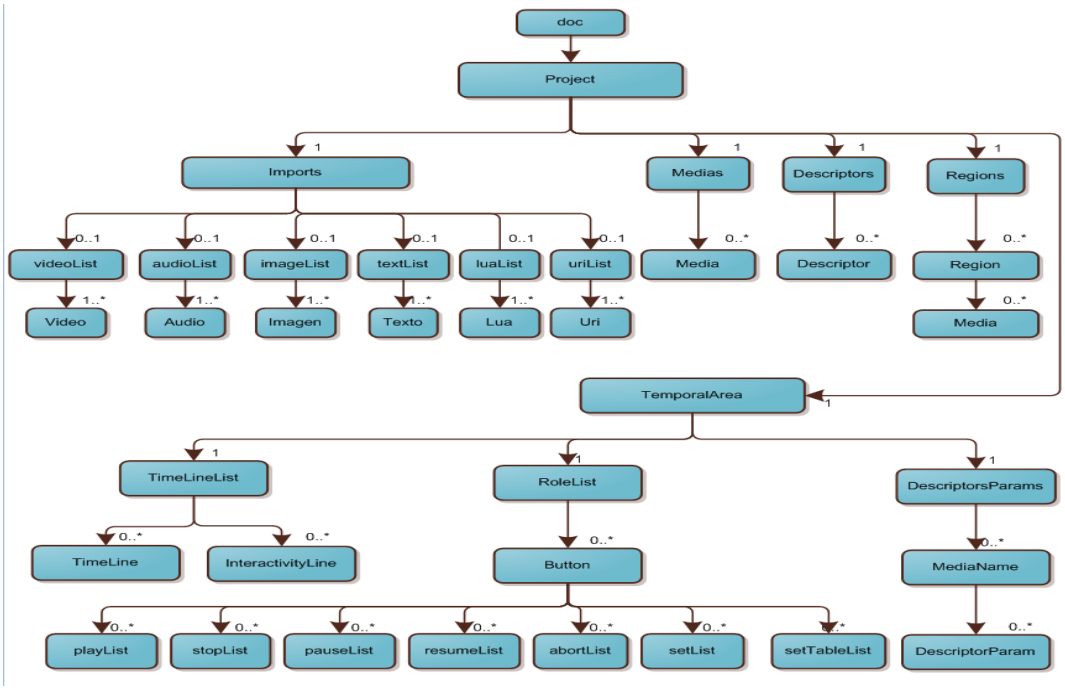


Figura 11: Modelo de objectos de dominio.

- `phonon`: Es la API multimedia de Qt. Por debajo esta biblioteca usa *gstreamer* o *ffmpeg* para la codificación, decodificación y streaming de video y sonido.
- `ffmeg`[15]: Conjunto de bibliotecas y utilidades para la codificación, decodificación, conversión, captura y streaming de audio y video. Este paquete también es libre y multiplataforma.
- `qmake`: Herramienta para generación de `Makefiles`.

En [7] se hace una descripción en profundidad de los detalles de diseño e implementación de la herramienta.

# 6. Conclusiones y trabajo futuro

*Crea TV Digital* permite la creación de aplicaciones interactivas de televisión digital brindando una interface visual simple e intuitiva, ocultando los detalles del lenguaje subyacente para el desarrollo de aplicaciones sobre la plataforma Ginga-NCL.

Su utilización aún no ha sido masivo pero ya se han recibido aportes de otros desarrolladores y se han reportado algunos errores y sugerencias de mejoras, lo cual muestra que ha causado interés en la comunidad de desarrolladores de aplicaciones para TV digital.

Es natural que esta versión inicial sufra de algunos defectos y posiblemente oculte algunos errores aún no detectados. A medida que se incremente su uso se irá perfeccionando y seguramente crecerá gracias a los aportes de la comunidad de desarrolladores de software libre.

Actualmente, *Crea TV Digital* se está utilizando en cursos de capacitación destinados a profesionales del área de producción audiovisual de la Universidad Nacional de Río Cuarto y se realizarán encuestas para recopilar experiencias en el uso y obtención de sugerencias por parte de los usuarios con el objetivo de ir mejorándola en cada nueva versión.

Entre las extensiones planificadas a corto plazo son la incorporación de:

- Mejores facilidades para el uso de scripts Lua durante el proceso de desarrollo.
- Mejora en la definición de alternativas de exhibición (`ver NCL <switch>`).
- Importación de proyectos desde archivos NCL existentes.
- Reestructuración del diseño del subsistema de generadores de código para soportar la *registración* en forma de *plug-ins*.
- Desarrollo de un instalador de binarios (ej: .deb para Debian/Ubuntu y setup para MS-Windows).
- Visualización de código NCL asociado a cada elemento del DOM, lo cual podría ser muy útil para la enseñanza de NCL.
- Desarrollo de un generador de código para HTML5/Javascript.

# Referencias


[1] *Normas Brasileiras de TV Digital*. `http://www.forumsbtvd.org.br/`

[2] Luis Fernando Gomes Soares, Simone Diniz Junqueira Barbosa. *Programando Em NCL*. Elsevier Editora Ltda. Rio de Janeiro. Brasil. ISBN 978-85-352-3457-2. 2009.

[3] Bertone, P; Wheeler, J.T.; Lujan,O.; Cocco,R.; Aramayo,A.; Boatti,A.; Ficco,O.;Perez,P.; Ammann,J. *Punto por punto. Sutura. Un recurso didactico como propuesta en la enseñanza de la cirugia veterinaria*. Presentación Poster en XI Seminario de Cirugia y taller de Enseñanza de Cirugia Veterinaria. Universidad Nacional del centro de la provincia de Buenos Aires. 2010.

[4] Bertone, P; Wheeler, J.T.; Verde,C.; Luján,O.; Cocco,R.; Rovere,R.; Aramayo,A.; Boatti,A. *Instrumental, nudos y ligaduras: una propuesta pedagogica a travès de un video hipermedia*. Presentación Poster en XII Seminario de Cirugia y taller de Enseñanza de Cirugia Veterinaria. Universidad Nacional de Tucumán. Horco Molle, Tucumán. 2011.

[5] Martinelli Fernán, Riberi Franco, Varea Agustín. *Suturas en Medicina Veterinaria: Un Prototipo de Desarrollo NCL para la plataforma de Televisión Digital*. Concurso de Trabajos Estudiantiles, EST, 40 JAIIO. Universidad Nacional de Córdoba. Septiembre de 2011.

[6] Sebastián Martín Schwartz , Sergio Alejandro Cardozo. *ÇreaTV Digital" - Herramienta para la creación de aplicaciones NCL para el middleware GINGA de Televisión Digital Terrestre.* Concurso de Trabajos Estudiantiles, EST, 40 JAIIO. Universidad Nacional de Córdoba. Septiembre de 2011.

[7] Sebastián Martín Schwartz , Sergio Alejandro Cardozo. Dir: Marcelo Arroyo. Codir: Laura Tardivo. *ÇreaTV Digital" - Herramienta para la creación de aplicaciones NCL para el middleware GINGA de Televisión Digital Terrestre.* Informe de tesina de Licenciatura en Ciencias de la Computación, FCEFQyN, Universidad Nacional de Río Cuarto. 2011.

[8] Roberto Ierusalimschy. *Programming in Lua*. Second Edition. ISBN: 978-8590379829. Lua.org. 2006.

[9] Sitio oficial de GINGA-AR. `http://ginga.org.ar`.

[10] NCL-Composer. `http://sourceforge.net/projects/composer-ncl/`.

[11] NCL-Eclipse plug-in. `http://laws.deinf.ufma.br/~ncleclipse/`.

[12] NCL-Eclipse plug-in. `http://www.eclipse.org/`.

[13] E. Gamma, R. Helm, R. Johnson, J. Vlissides. *Design Patterns. Elements of Reusable Object-Oriented Software*. Addison Wesley. ISBN-10: 0201633612, ISBN-13: 978-0201633610. 1994.

[14] Qt framework. `http://qt.nokia.com/products/`.

[15] FFMPEG. `http://ffmpeg.org`.

# Anexos

## A. Ejemplo de uso de Crea TV Digital

En este anexo se desarrollará una aplicación paso a paso usando *Crea TV Digital*. El objetivo de la aplicación es mostrar las características de la definición visual de objectos multimedia relacionados espacialmente y temporalmente con inicio de exhibición de objetos multimedia mediante la ocurrencia de eventos temporales y/o eventos causados por interactividad generada por el usuario.

La aplicación desarrollada hará lo siguiente:

1. Comienza con la exhibición de un video principal en una región que cubre la pantalla completa.
2. Al inicio del video principal también se exhibe una imagen con un botón rojo, indicando que está habilitada la interactividad (indicando al usuario que presionando el botón rojo del control remoto se iniciará alguna actividad).
3. En el momento que el usuario pulse el botón rojo del control remoto, se inicia la reproducción de un video en la esquina superior derecha de la pantalla.
4. Al finalizar este video, el botón rojo desaparecerá y se mostrará un botón verde.
5. Cuando el usuario pulse el botón verde, se reemplazará el video principal con una imagen de un paisaje y se dará comienzo a la reproducción de otro video en la esquina superior derecha de la pantalla.

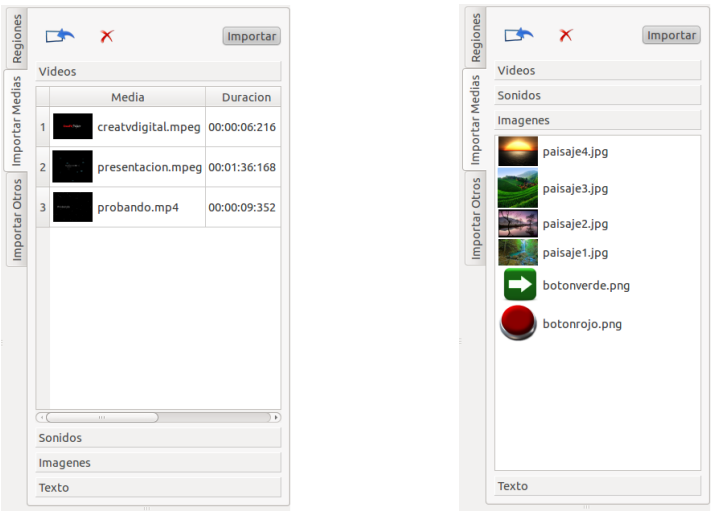


Figura 12: Importación de objetos multimedia.

A continuación se muestran los pasos a realizar en la interface de *Crea TV Digital*.

1. **Importación de objetos multimedia:** Seleccionar los objetos multimedia a utilizar. La figura 13 muestra las pantallas resultantes:
2. **Creación de regiones:** y sus vinculaciones con objetos multimedia (ver figura 13).
3. **Inicio de la aplicación:** Crear la primera línea de tiempo para el objeto (video *presentacion.mpeg*) a reproducir inicialmente (ver figura 14).

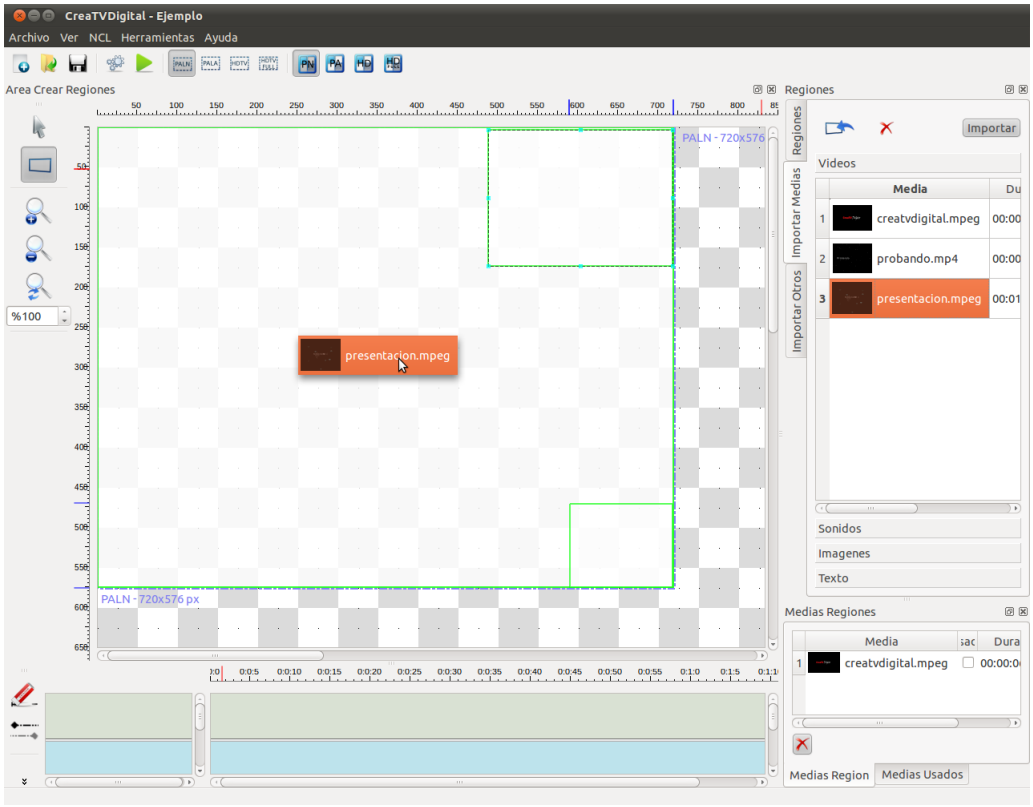


Figura 13: Regiones relacionadas a objetos multimedia.

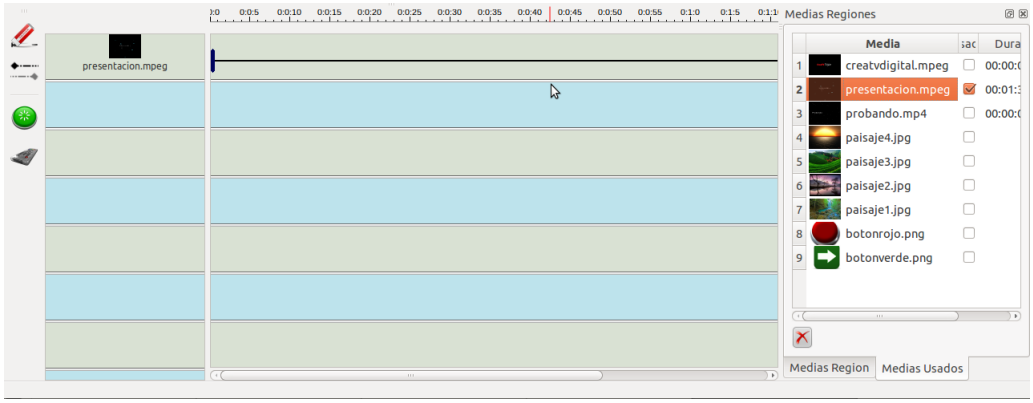


Figura 14: Objeto multimedia de reproducción al inicio.

4. **Interactividad (botón rojo):** Crear línea de interactividad del botón rojo (previamente adicionado con su correspondiente línea de tiempo) asociado a la reproducción del video en la esquina superior derecha de la pantalla (ver figura 15).

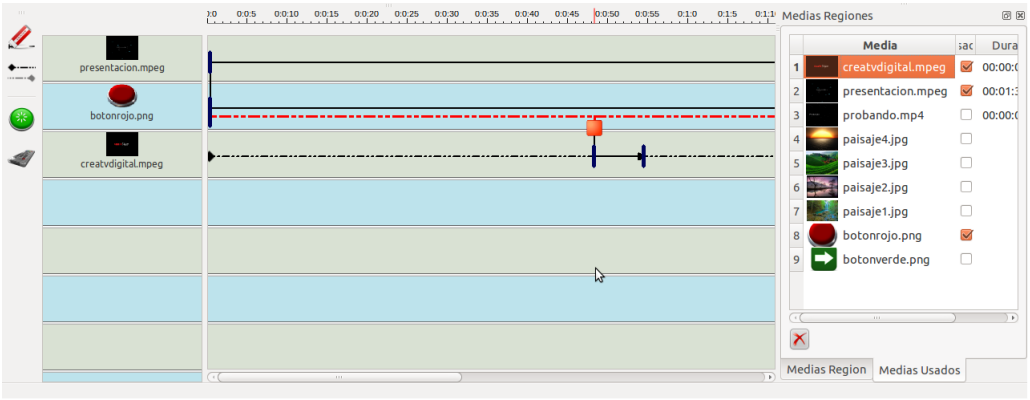


Figura 15: Creación de línea de interactividad.

5. **Interactividad (botón verde):** Crear línea de interactividad del botón verde asociado al evento de finalización del video iniciado en el paso anterior y la reproducción del video en la esquina superior derecha de la pantalla (fig. 15). Además crear la última línea de interactividad comenzando de este mismo instante de tiempo y asociarlo con las acciones de parar la reproducción del video principal y reemplazarlo por la imagen de un paisaje. La figura 16 muestra el botón verde de interactividad con las dos acciones asociadas (parar `presentacion.mpeg` y comenzar `paisaje1.png`).

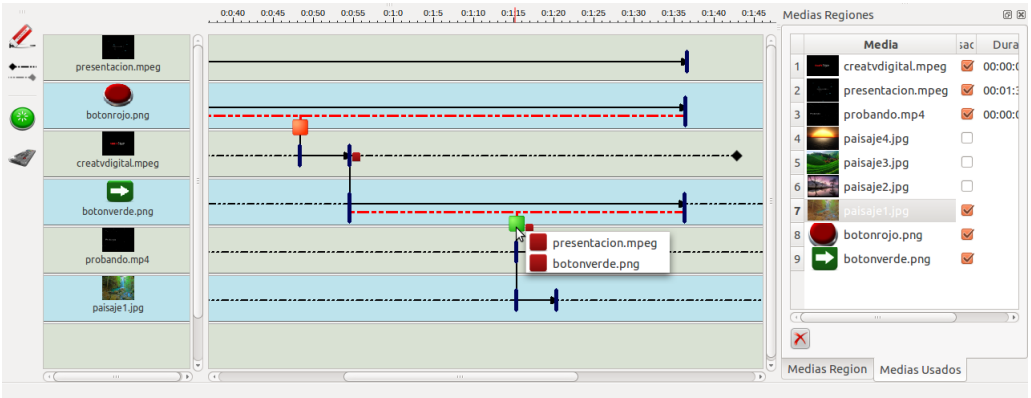


Figura 16: Interactividad botón verde .

6. **Generación de código NCL:** Pulsar el botón de generación de código ( ) en la barra de herramientas.
7. **Generación de código NCL:** Pulsar el botón de ejecución de la aplicación ( ) en la barra de herramientas.

La figura 17 muestra la aplicación resultante corriendo con ginga.

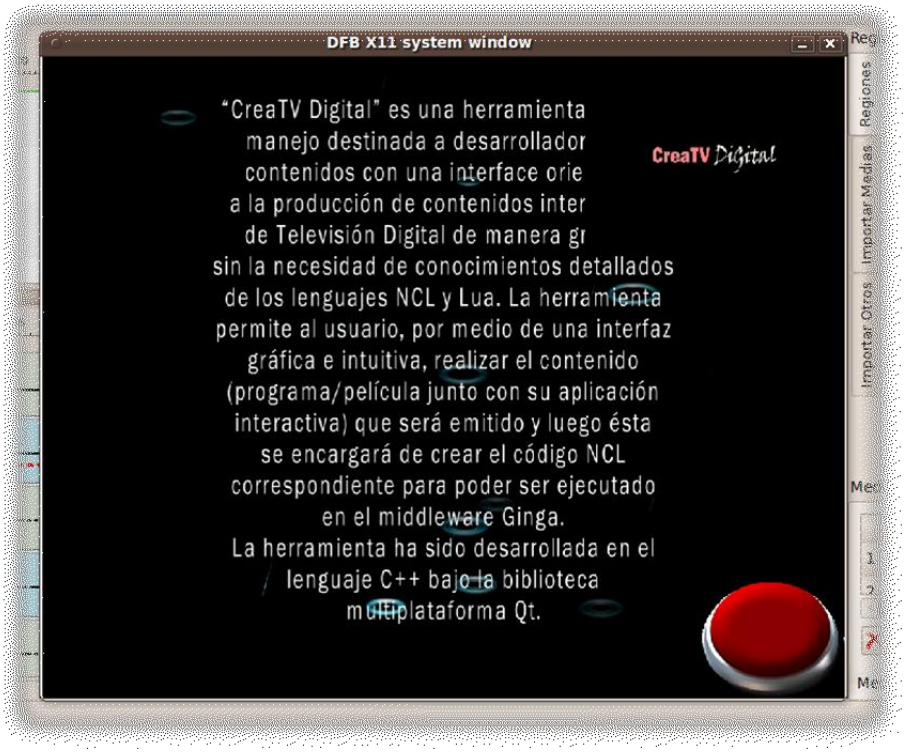


Figura 17: Aplicación generada corriendo con `ginga`.

# B. Compilación e instalación de Crea TV Digital

Es este anexo se describe el proceso de compilación e instalación de *Crea TV digital*.

1. Instalar las herramientas de desarrollo: compilador `g++` y utilidad `make`.
2. Descargar *Crea TV digital* de `http://code.google.com/p/creatvdigital/`.
3. Instalación de las dependencias.
   *a*) Biblioteca de desarrollo `Qt` versión 4. En distribuciones Ubuntu el paquete se denomina `libqt4-dev`.
   *b*) Biblioteca `phonon`. En Ubuntu es el paquete `libphonon-dev`.
   *c*) Ffmpeg: `libxine1-ffmpeg` y paquete `ffmpeg`.
4. Descomprimir CreaTVDigital.tar.gz.
5. En la carpeta descomprimida, realizar:
   *a*) `qmake` (crea el archivo Makefile)
   *b*) `make` (compila los archivos fuentes y genera el binario *CreaTVDigital*)

El binario se genera en la carpeta de compilación. Si se desea instalarlo para todos los usuarios del sistema copiar el binario en alguna carpeta de binarios (como `/usr/local/bin/` en GNU-Linux).