

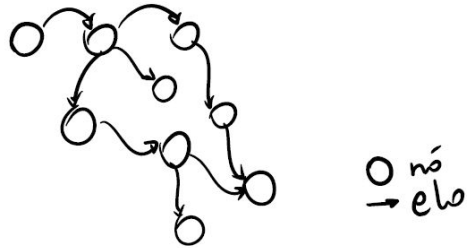
Pós-Graduação
MDD – Mídias Interativas
Ginga NCL 3.0



Prof.^a Graciana Simoní Fischer de Gouvêa

GINGA - NCL 3.0

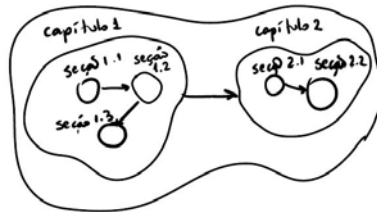
- ✓ Documentos hipermídia são geralmente compostos de nós (*nodes*) e elos (*links*)



- ✓ NCL = *Nested Context Language*
- ✓ Possibilita a elaboração de documentos hipermídia, com sincronismo entre mídias e interação com usuário.

GINGA - NCL 3.0

- ✓ Baseado no NCM = *Nested Context Model* ou Modelo de Contextos Aninhados (nó de composição)



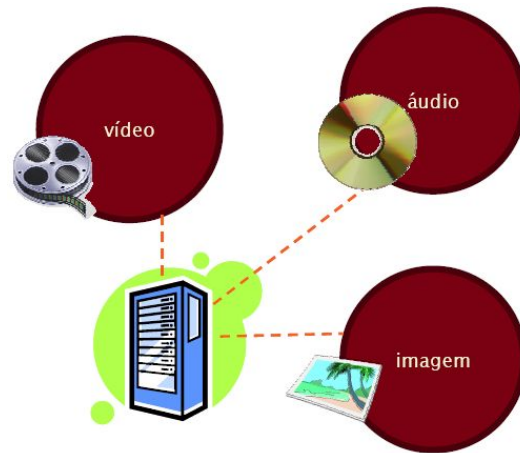
- ✓ Um *node* (nó) pode ser de dois tipos:
 - Nó de conteúdo ou de mídia (*content node* ou *media node*): associando a um elemento de mídia como vídeo, áudio, imagem, texto, aplicação, etc., ou
 - Nó de composição ou contexto (*composite node* ou *context*, que utiliza um *switch*).

GINGA - NCL 3.0

- ✓ Para construir um documento hipermídia, é necessário definir:
 - **O que** se quer tocar;
 - **Onde** (i.e. em que região da tela e de qual dispositivo);
 - **Como** (i.e. em que volume, com que *player*, etc);
 - **Quando** (antes/depois de qual mídia ser apresentada ou após qual tecla ser pressionada).

O que tocar?

- ✓ A primeira coisa que deve-se considerar é o conteúdo.
- ✓ O conteúdo é representado através de **nós de mídia**.



Em NCL, um nó de mídia ou de conteúdo (*media node* ou *content node*) define o objeto de mídia propriamente dito: vídeo, áudio, texto, imagem, etc. Cada definição de nó de mídia deve apresentar além do arquivo de origem da mídia, o descritor que regulará a apresentação daquele objeto.

Um nó de mídia <media node> pode ter os seguintes atributos:

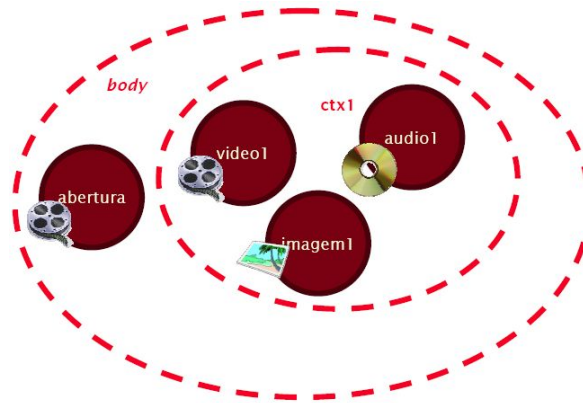
- **id***: identificador único, que é utilizado para se referir ao nó.
- **type**: tipo de mídia que pode ser: (htm/html, css, xml, bmp, png, gif, jpg, wav, mp3, mp2, mp4/mpg4, mpeg/mpg, lua, xlt/xlet/class, x-ginga-settings (nó de atributos globais para ser utilizado em regras e *switches*) e x-ginga-time).
- **src**: fonte do objeto de mídia, caminho para o arquivo (absoluto ou relativo)
- **descriptor**: descritor que controla a apresentação do objeto de mídia.
- **refer**: referência a um outro nó de mídia previamente definido, como forma de reuso de nó.
- **newInstance**: define se um nó que se refere a outro gera uma nova instância do objeto ou se utiliza a instância previamente criada.

Exemplo:

```
<media id="video1" type="video/mpeg" src="media/video1.mpg" descriptor="dVideo" />
```

O que tocar?

- ✓ Todo **nó de mídia** é definido dentro de um contexto.
- ✓ O elemento *body* é o contexto que contém todos os nós do documento, sejam um nó de mídia ou contexto.



Os contextos são utilizados para estruturar um documento hipermídia, podendo ser aninhados e refletindo assim a estrutura do documento, ajudando a organizar os segmentos do programa audiovisual interativo.

O elemento *body* é um caso particular de contexto, representando o documento como um todo.

Os atributos de um contexto são:

id*: identificador único do contexto.

- **refer**: referência a um outro contexto previamente definido.

É definido da seguinte forma:

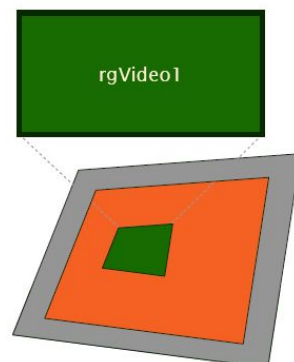
```
<context id="ctxName">
```

```
...
```

```
</context>
```

Onde tocar?

- ✓ À medida em que se define o conteúdo, define-se também a área em que este será apresentado na tela, através de elementos denominados **regiões**.
- ✓ Uma região indica a posição e as dimensões da área onde uma mídia será apresentada.



Em NCL, as regiões são definidas no cabeçalho do programa <head> na seção de base de regiões <regionBase>. Todo documento NCL possui pelo menos uma região, que define a dimensão e as características do dispositivo onde um ou mais nós de mídia serão apresentados.

Uma região base <regionBase> pode ter os seguintes atributos:

id: identificador único

device: dispositivo de associação da região.

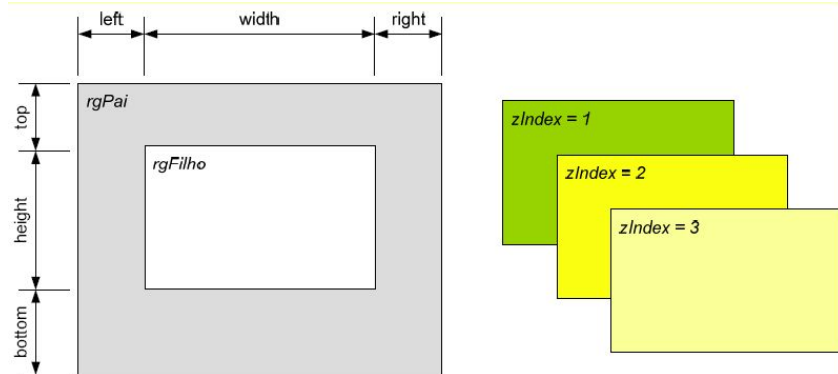
Exemplo:

```
<regionBase>
```

```
  <region id="rgVideo" height="240" width="320" />
```

```
</regionBase>
```

Onde tocar?



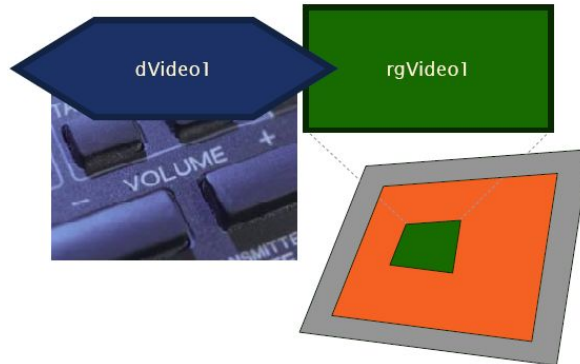
Uma região <region> pode ter os seguintes atributos:

- **id***: identificador único, que é utilizado para se referir à região.
- **title**: indicado para o caso da região ser exibida com uma moldura.
- **height**: altura da região em pixels.
- **width**: largura da região em pixels.
- **left**: distância em pixels em relação à borda esquerda da tela ou da região pai.
- **top**: distância em pixels em relação à borda superior da tela ou região pai.
- **right**: distância em pixels em relação à borda direita da tela ou da região pai.
- **bottom**: distância em pixels em relação à borda inferior da tela ou da região pai.
- **zIndex**: usado em caso de sobreposição de regiões - maior fica em cima.

Obs.: Os atributos **left** e **width** têm precedência sobre o atributo **right**, assim como os atributos **top** e **height** têm precedência sobre o atributo **bottom**.

Como tocar?

- ✓ A associação entre uma mídia e sua região é feita através de um **descriptor**.
- ✓ É utilizado também para definir a forma como a mídia será apresentada, o volume, transparência, etc.



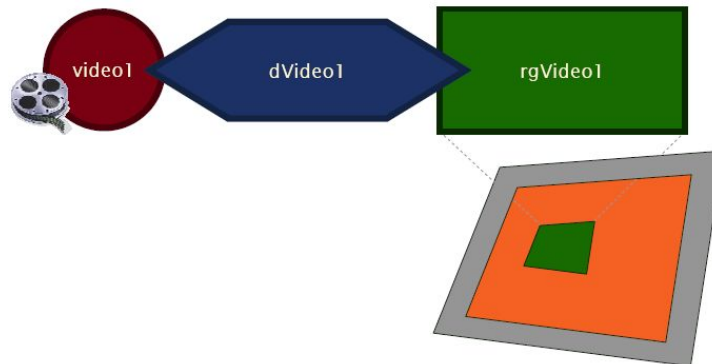
Em NCL, todos os descritores são definidas no cabeçalho do programa <head> na seção de base de descritores <descriptorBase>. Todo nó de mídia que será apresentado deve ter um descriptor associado.

Um descriptor <descriptor> pode ter os seguintes atributos:

- **id***: identificador único, que é utilizado para se referir ao descriptor.
- **player**: (opcional) ferramenta que será utilizada para exibir a mídia associada ao descriptor.
- **explicitDur**: duração ideal da mídia associada ao descriptor. Deve ser expresso em segundos no formato 9.9s. Qdo não especificado, a mídia é exibida com seu valor *default*. Mídias de texto e imagens têm duração *default* infinita. Em mídias contínuas, não há necessidade.
- **region**: região associada ao descriptor, ligando a mesma ao objeto de mídia (visual).
- **freeze**: o que acontecerá ao terminar a mídia. Ex.: *true* em um vídeo, congela o último frame.
- **focusIndex**: índice de navegação para o objeto de mídia associado ao descriptor. O foco inicial estará no objeto com **focusIndex** menor. Em caso de não ser número, é a menor palavra.
- **focusBorderColor**: cor do retângulo que deve aparecer sobrescrito à região do mesmo, quando o objeto de mídia a ele associado receber foco. Valor: *white, black, silver, gray, red, maroon, fuchsia, purple, lime, green, yellow, olive, blue, navy, acqua* ou *teal*.
- **focusBorderWidth**: espessura da borda em pixels. Valor 0, não exibe a borda. Valor positivo indica que a borda ficará fora do objeto (externo) e valor negativo, borda sobre o objeto (interno).
- **focusBorderTransparency**: porcentagem de transparência da borda. Valor entre 0 e 1, onde 0 significa totalmente opaco e 1 totalmente transparente.
- **focusSrc**: mídia alternativa a ser exibida quando a mídia associada ao descriptor estiver com foco.

Como tocar?

- ✓ Mesmo que se queira alterar a forma de apresentação da mídia, é necessário um descritor para **associar a mídia à região**.



- **focusSelSrc**: mídia alternativa a ser exibida quando o botão OK ou Enter for pressionado, enquanto a mídia associada ao descritor estiver com o foco.
- **selBorderColor**: cor de borda a ser exibida quando o botão OK ou Enter for pressionado, enquanto a mídia associada ao descritor estiver com o foco.
- **moveLeft**: índice de navegação do elemento *E* que deve receber foco caso seja pressionada a seta para a esquerda do controle remoto ou enquanto a mídia associada ao descritor estiver com foco (definido pelo atributo **focusIndex** do elemento *E*).
- **moveRight**: índice de navegação do elemento *E* que deve receber foco caso seja pressionada a seta para a direita do controle remoto ou enquanto a mídia associada ao descritor estiver com foco (definido pelo atributo **focusIndex** do elemento *E*).
- **moveUp**: índice de navegação do elemento *E* que deve receber foco caso seja pressionada a seta para cima do controle remoto ou enquanto a mídia associada ao descritor estiver com foco (definido pelo atributo **focusIndex** do elemento *E*).
- **moveDown**: índice de navegação do elemento *E* que deve receber foco caso seja pressionada a seta para baixo do controle remoto ou enquanto a mídia associada ao descritor estiver com foco (definido pelo atributo **focusIndex** do elemento *E*).
- **transIn**: transição que será executada ao iniciar a apresentação da mídia associada ao descritor.
- **transOut**: transição que será executada ao terminar a apresentação da mídia associada ao descritor.

Emento opcional:

- **descriptorParam**: parâmetro do descritor como um par <propriedade, valor> que dependem do programa de exibição da mídia associada ao descritor. Cada descritor pode conter diversos:
`<descriptorParam name="nome_do_parâmetro" value="valor_do_parâmetro" />`

Como tocar?

- ✓ O uso de parâmetros de descritor promove um alto grau de flexibilidade:

```
<descriptor id="dVideo" region="rgVideo" >
  <descriptorParam name="soundLevel" value="0,9s" />
</descriptor>
```

- ✓ Cabe a cada programa de exibição do objeto de mídia <player> interpretar essas propriedades de forma adequada.
- ✓ Atualmente não é possível definir parâmetros nas janelas do **Composer**, apenas diretamente no código.

Parâmetros de descritor:

Tipo de mídia: objetos com áudio

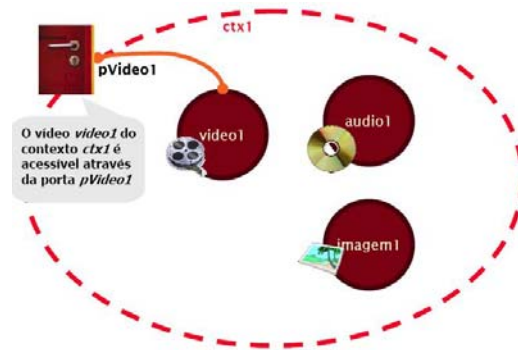
soundLevel balanceLevel trebleLevel bassLevel	Valores entre 0 e 1. No caso de soundLevel, 0 = mute; 0.5 = volume a 50%; e 1 = volume máximo.
--	---

Tipo de mídia: Objetos visuais

location	Posição: <left,top> valores 0-100% ou em pixels.
size	Dimensões: <width,height> valores 0-100% ou em pixels.
bounds	Posição e dimensão. <left,top,width,height> valores 0-100% ou em pixels.
background	Nomes de cores reservados e transparent para o caso de Gif's.
visible	true ou false.
transparency	0 (opaco) ou 1 (transparente)
fit	fill: redimensiona conteúdo do objeto de mídia para q. toque as bordas da região. hidden: se altura menor que height, renderizado a partir do topo e altura restante preenchida com cor fundo, se maior, restante deve ser cortado. Idem para a largura e esquerda. meet: redimensiona mantendo proporções até atingir uma das bordas da região. Vazios direita ou inferior, devem ser preenchidos com cor de fundo. meetBest: semelhante ao meet, mas objeto não é renderizado mais que dobro sua dimensão. Slice: redimensiona mantendo proporções até toda região preenchida. Pode cortar conteúdo.
scroll	none, horizontal, vertical, both ou automatic
style	localização de um arquivo de folha de estilo

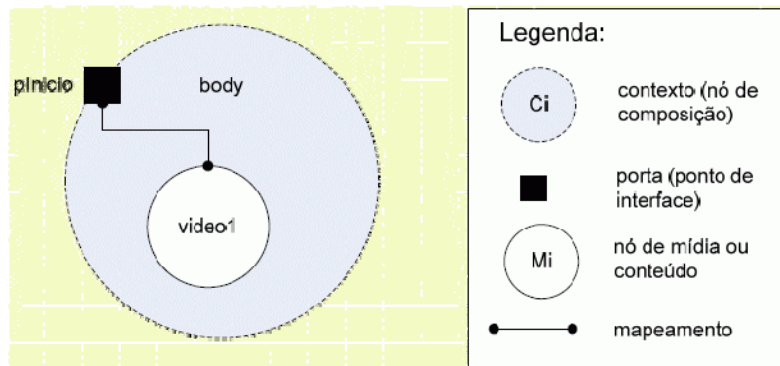
Quando tocar?

- ✓ Para definir o **primeiro nó** do documento:
 - Deve-se criar uma **porta de contexto** *body* para este nó.
- ✓ Uma porta identifica por onde o documento deve começar, assim como são necessárias portas para dar acesso aos nós de fora do seu contexto.



Uma **porta** é um ponto de interface (*interface point*) de um contexto, que oferece acesso externo ao conteúdo (nós internos) do mesmo. Ou seja, para um elo apontar para um nó interno ao contexto, este deve possuir uma porta que leve ao nó interno desejado.

Em todo documento deve haver ao menos uma porta de entrada (port na seção body).



Uma porta possui os seguintes atributos:

- **id***: identificador único, utilizado nas referências à porta (por exemplo, nos elos).
- **component***: nó de mídia ou contexto que a porta mapeia.
 - Se for contexto, deve-se definir o atributo **interface**, fazendo o mapeamento para uma porta ou âncora daquele contexto.
 - Se for nó de mídia, pode-se definir o atributo **interface**, apenas como sendo uma âncora do nó de mídia. Omitindo, todo o nó é considerado mapeado para aquela porta.
- **interface**: ponto de interface no contexto ou âncora de destino no nó de mídia ou contexto.

Exemplo:

```
<port id="port_nomeDocumento" component="video1"/>
```

Quando tocar?

- ✓ Para definir **quando** um nó de mídia será apresentado em relação aos outros, são criados **elos**.
- ✓ **Os elos** são utilizados para:
 - Estabelecer o **sincronismo entre os nós** e
 - Definir a interatividade do programa.
- ✓ O comportamento destes elos é dado por **conectores**.

Os elos (*links*) associam nós através de conectores (*connectors*) que definem a semântica da associação entre os nós. A NCL define os seguintes atributos de elos:

- **id***: identificador único do elo.
- **xconnector***: identificador do conector associado ao elo.

A NCL define os seguintes elementos contidos num elemento de elo:

- **linkParam**: define um parâmetro do elo como um par <propriedade, valor>. As propriedades e seus respectivos valores *dependem da definição do conector ao qual o elo está associado*. Um elo pode conter diversos elementos **linkParam**.
- **bind**: indica um componente (*component*, nó de mídia ou de contexto) envolvido no elo, indicando seu papel (*role*) no elo, *conforme a semântica do conector*. Em alguns casos deve-se indicar também o ponto de interface (*interface*) do nó ao qual o elo é ligado (porta do contexto ou âncora de um nó de mídia). Um elo pode conter diversos elementos **bind**, e deve conter pelo menos um bind para cada papel definido no conector.

O elemento **bind** pode ainda conter uma ou mais instâncias de parâmetros como elementos filhos:

- **bindParam**: define um parâmetro específico do bind como um par <propriedade, valor>. As propriedades e seus respectivos valores dependem da definição do conector ao qual o elo está associado.

Conectores

- ✓ Definem o **sincronismo** por mecanismos de causalidade e restrição, através de **papéis (roles)** que os nós de origem e destino exercem nos **elos** que utilizam aquele conector

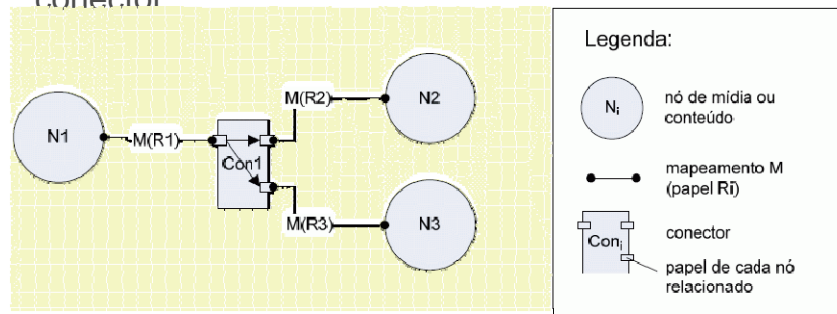


Imagem de um conector ligando três nós

No NCM e na NCL, o sincronismo não é feito por marcas de tempo (*timestamps*), mas por mecanismos de causalidade e restrição definidos nos conectores (*connectors*). O conector define os papéis (*roles*) que os nós de origem e destino exercem nos elos que utilizam o conector. Na NCL 3.0 existe apenas um tipo de conector: o conector causal (*causal connector*). Ele define condições (*condition*) sob as quais o elo pode ser ativado, e as ações (*action*) que serão realizadas quando o elo for ativado. Um conector causal deve possuir ao menos uma condição e uma ação. Cada condição ou ação é associada a um papel (*role*), ponto de interface que participa dos mapeamentos do elo.

Papéis de condição pré-definidos:

Papel	descrição (o elo será ativado quando...)
onBegin	...a apresentação do nó de mídia associado ao papel onBegin for iniciada
onEnd	...a apresentação do nó de mídia associado ao papel onEnd terminar ou por stop
onAbort	...a apresentação do nó de mídia associado ao papel onAbort for abortada
onPause	...a apresentação do nó de mídia associado ao papel onPause for pausada
onResume	...a apresentação do nó de mídia associado ao papel onAbort for retomada (após pause)
onSelection	...uma tecla (key) for pressionada, ou a tecla OK for pressionada qdo nó de mídia tiver foco
onAttribution	...um valor <value> for atribuído

Papéis de ação pré-definidos

Papel	descrição (quando o elo for ativado...)
start	...inicia a apresentação do nó de mídia associado ao papel start
stop	...termina a apresentação do nó de mídia associado ao papel stop
abort	...aborta a apresentação do nó de mídia associado ao papel abort
pause	...pausa a apresentação do nó de mídia associado ao papel pause
resume	...retoma a apresentação do nó de mídia associado ao papel resume (caso esteja em pause)
set	...estabelece um valor <value> à âncora associada ao papel set

Conectores

- ✓ Tanto os papéis de condição quanto os de ação estão associados a transições de estados numa máquina de eventos:

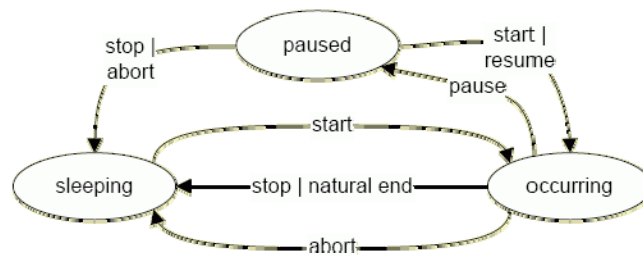
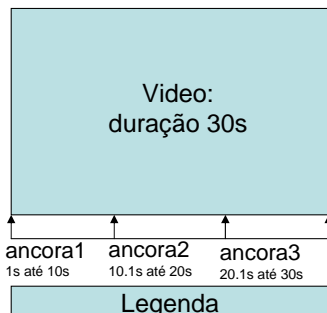


Imagem da máquina de estado de eventos

Alguns dos principais conectores estão mapeados nos ANEXOS.

Âncoras

- ✓ Pontos de entrada para os nós de mídia ou contextos.
- ✓ Seu objetivo é mapear **segmentos** ou **propriedades**, seja como origem ou destino de elos.
- ✓ Podem ser âncoras de **conteúdo** (*content anchor*) ou âncoras de **propriedade** (*property anchor*)



Legenda1

Ocorrerá durante 10s no início do vídeo
Então deverá ser ligada ao vídeo (âncora1)

Legenda2

Ocorrerá durante 10s, depois da legenda1
Então deverá ser ligada ao vídeo (âncora2)

Legenda3

Ocorrerá durante 10s, depois da legenda2
Então deverá ser ligada ao vídeo (âncora3)

Exemplo de âncora de conteúdo: vídeo com 3 legendas

Âncora de conteúdo

Define um segmento da mídia (intervalo de tempo e/ou região no espaço) que poderá ser utilizado como ponto de referência de elos. Um segmento de mídia é uma seleção contínua de **unidades de informação** (*information units*) de um nó. Sua definição depende do tipo de mídia representado pelo nó. No caso do vídeo, por exemplo, poderiam ser *frames* do vídeo.

É definida como um elemento **area** dentro do elemento **media**, como por exemplo:

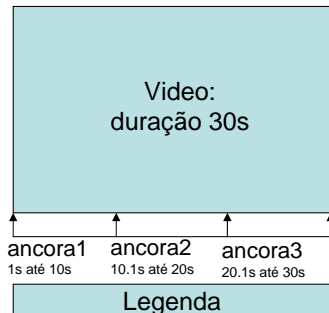
```
<media type="video" id="video1" src="media/video1.mpg" descriptor="dVideo">  
<!--âncoras de conteúdo no vídeo que devem ser sincronizadas com a legenda -->  
  <area id="ancora1" begin="1s" end="10s"/>  
  <area id="ancora2" begin="10.1s" end="20s"/>  
  <area id="ancora3" begin="20.1s" end="30s"/>  
</media>
```

Uma âncora de conteúdo pode ter os seguintes atributos:

- **id***: identificador único da âncora.
- **coords**: coordenadas em pixels da âncora espacial (mídias visuais), formato "X,Y,width,height".
- **begin**: início da âncora, em segundos, no formato "99.9s" (mídias contínuas)
- **end**: término da âncora, em segundos, no formato "99.9s" (mídias contínuas)
- **dur**: duração da âncora, em segundos, no formato "99.9s" (mídias contínuas)
- **first**: quadro/amostra da mídia, definindo o início da âncora (mídias contínuas)
- **last**: quadro/amostra da mídia, definindo o fim da âncora (mídias contínuas)
- **text**: texto da âncora no arquivo de origem (mídias textuais)
- **position**: posição do texto da âncora a partir do arquivo de origem (mídias textuais)
- **anchorLabel**: identificador da âncora no arq. de origem, como interpretado pela *tool* de exibição.

Âncoras

- ✓ Pontos de entrada para os nós de mídia ou contextos.
- ✓ Seu objetivo é mapear **segmentos** ou **propriedades**, seja como origem ou destino de elos.
- ✓ Podem ser âncoras de **conteúdo** (*content anchor*) ou âncoras de **propriedade** (*property anchor*)



Legenda1
Ocorrerá durante 10s no início do vídeo
Então deverá ser ligada ao vídeo (âncora1)

Legenda2
Ocorrerá durante 10s, depois da legenda1
Então deverá ser ligada ao vídeo (âncora2)

Legenda1
Ocorrerá durante 10s, depois da legenda2
Então deverá ser ligada ao vídeo (âncora3)

Exemplo de âncora de conteúdo: vídeo com 3 legendas

Âncora de propriedade




Referem-se a propriedades de um nó de origem ou de destino, que podem ser manipuladas pelos elos. Exemplos: volume de áudio de um nó de áudio ou vídeo, coordenadas e dimensões de exibição de um nó de mídia visual, etc.

É definida como um elemento **property** dentro do elemento **media** ou **context**, como no exemplo, em que são definidas quatro âncoras de propriedade para um nó de vídeo, além de uma âncora de conteúdo:

```
<media type="video" id="video1" src="media/video1.mpg" descriptor="dVideo">  
  <!--âncoras de propriedade que serão manipuladas pelos elos e/ou descritores-->  
  <property name="top" />  
  <property name="left" />  
  <property name="width" />  
  <property name="height" />  
  
  <!--âncora de conteúdo no vídeo que deve sincronizar com a legenda -->  
  <area id="ancora1" begin="3s" end="8s" />  
</media>
```

Estrutura de um documento NCL

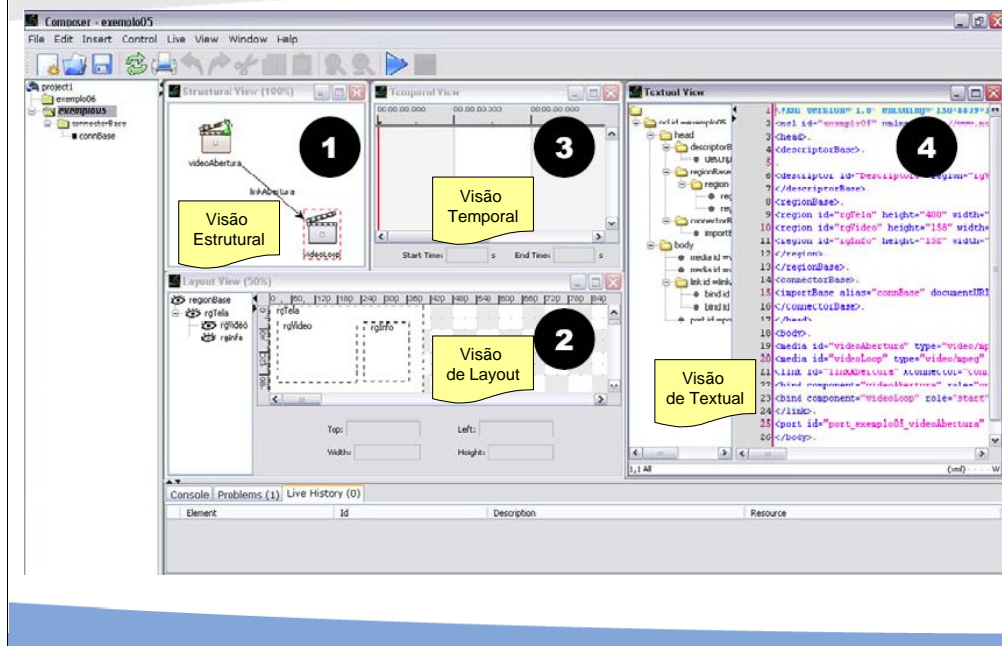
- ✓ Um documento NCL é um arquivo escrito em XML.
 - Um **cabeçalho** NCL (**linhas 1 e 2**)
 - Uma **seção do cabeçalho** do programa (seção *head*, **linhas 3 a 13**), onde se definem as **regiões**, **descritores**, **conectores** e as regras utilizadas pelo programa;
 - O corpo do programa (seção *body*, **linhas 14 a 17**) onde se definem os **contextos**, **nós de mídia**, **elos** e outros elementos que definem o conteúdo e estrutura do programa.
 - Pelo menos uma porta que indica por onde o programa começa a ser exibido (*port ptInicio*, **linha 15**) e
 - A conclusão do documento (**linha 18**)

cabeçalho do arquivo NCL	<p>1: <?xml version="1.0" encoding="ISO-8859-1"?></p> <p>2: <ncl id="exemplo01" xmlns="http://www.ncl.org.br/NCL3.0/EDTVProfile" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.ncl.org.br/NCL3.0/EDTVProfile http://www.ncl.org.br/NCL3.0/profiles/NCL30EDTV.xsd"></p>	1
cabeçalho do programa	3: <head>	
base de regiões	<p>4: <regionBase></p> <p>5: <!-- regiões da tela onde as mídias são apresentadas --></p> <p>6: </regionBase></p>	 2
base de descritores	<p>7: <descriptorBase></p> <p>8: <!-- descritores que definem como as mídias são apresentadas --></p> <p>9: </descriptorBase></p>	 3
base de conectores	<p>10: <connectorBase></p> <p>11: <!-- conectores que definem como os elos são ativados e o que eles disparam --></p> <p>12: </connectorBase></p>	8
	13: </head>	
corpo do programa	14: <body>	
ponto de entrada no programa	15: <port id="plnicio" component="ncPrincipal" interface="ilnicio"/>	5
conteúdo do programa	16: <!-- contextos, nós de mídia e suas âncoras, elos e outros elementos -->	 4
	17: </body>	6 7
término	18: </ncl>	

Estrutura de um documento NCL

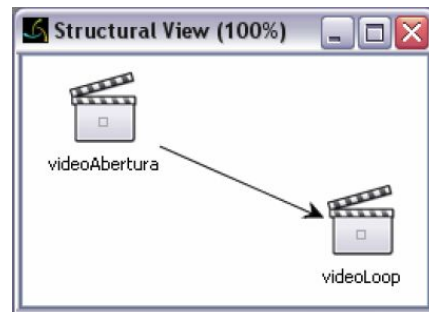
- ❶ os cabeçalhos básicos do arquivo NCL e do programa;
 - ❷ as regiões da tela onde aparecerão os elementos visuais (**regionBase**);
 - ❸ como e onde os nós de mídia serão exibidos, através de descritores (**descriptorBase**);
 - ❹ o conteúdo (nós de mídia - **media**) e a estrutura (contextos - **context**) do documento (seção **body**), associando-os aos descritores;
 - ❺ a porta de entrada do programa, apontando para o primeiro nó a ser exibido, bem como as portas para os contextos, visando à construção dos elos entre contextos e nós de mídia (**port**);
 - ❻ âncoras para os nós de mídia, visando à construção dos elos entre nós (**area** e **attribute**);
 - ❼ elos para o sincronismo e interatividade entre os nós de mídia e contextos (**link**); e
 - ❽ os conectores que especificam o comportamento dos elos do documento (**connectorBase**).
- Os elementos 1 a 5 são apresentados no exemplo 1.*
- Âncoras são vistas no exemplo 3.*
- Elos e conectores são vistos no exemplo 2.*

Ferramenta Composer



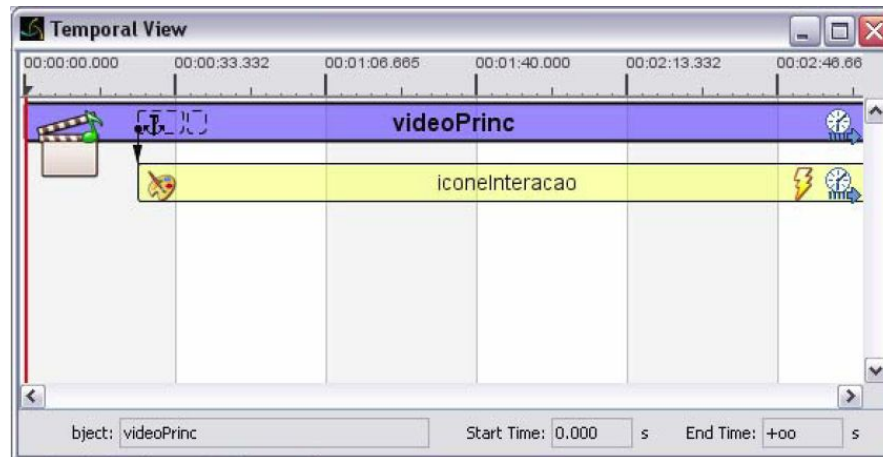
Composer – Visão Estrutural

- ✓ Apresenta os nós e elos entre os nós. Nesta visão é possível criar nós de mídia, contextos e elos, bem como definir suas propriedades



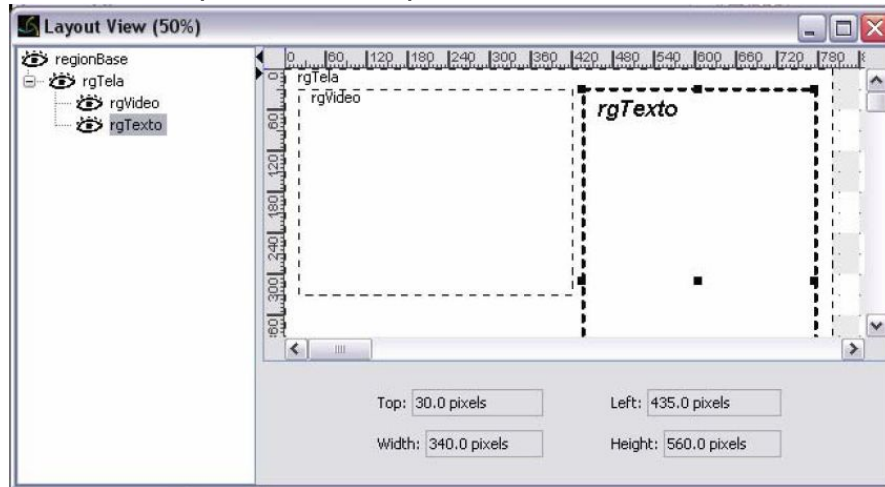
Composer – Visão Temporal

- ✓ Ilustra o sincronismo temporal entre os nós de mídia, e as oportunidades de interatividade.



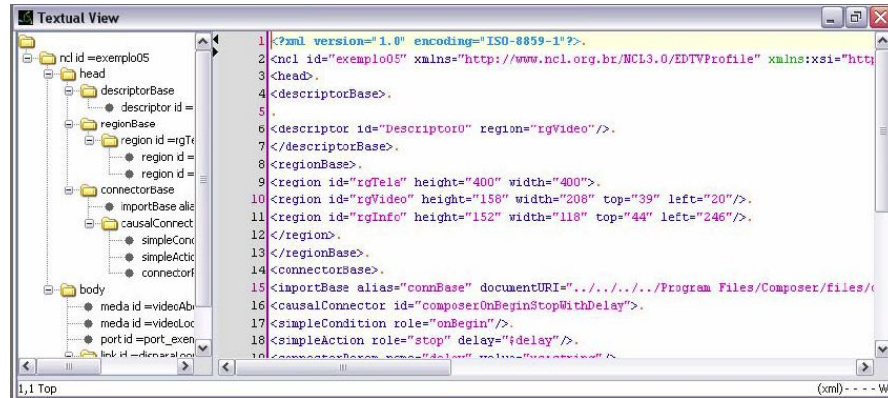
Composer – Visão de Layout

- ✓ Apresenta as regiões da tela onde as mídias do documento poderão ser apresentadas.



Composer – Visão Textual

- ✓ Apresenta o código NCL em si, podendo ser editado diretamente, refletindo as mudanças nas demais visões.



The screenshot shows a window titled "Textual View" with a tree view on the left and XML code on the right. The tree view shows a hierarchy starting with "nd id = exemplo05", followed by "head", "descriptorBase", "regionBase", "region id = rgTele", "connectorBase", "importBase", "causalConnect", "simpleConc", "simpleActio", and "connector". The XML code on the right is as follows:

```
1 <?xml version="1.0" encoding="ISO-8859-1"?>
2 <ncl id="exemplo05" xmlns="http://www.ncl.org.br/NCL3.0/EDTVProfile" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
3 <head>
4 <descriptorBase>
5 .
6 <descriptor id="Descriptor0" region="rgVideo"/>
7 </descriptorBase>
8 <regionBase>
9 <region id="rgTele" height="400" width="400">
10 <region id="rgVideo" height="158" width="208" top="39" left="20"/>
11 <region id="rgInfo" height="152" width="118" top="44" left="246"/>
12 </region>
13 </regionBase>
14 <connectorBase>
15 <importBase alias="connBase" documentURI="../../../../../../Program Files/Composer/files/connBase.ncl"/>
16 <causalConnector id="composerOnBeginStopWithDelay">
17 <simpleCondition role="onBegin"/>
18 <simpleAction role="stop" delay="{delay}"/>
19 </connectorBase>
20 </ncl>
```

Ferramentas requisitadas

- ✓ Instalação da versão Java do Composer, que já vem com o gingaNclPlayer.
 - Disponível para download em:
 - http://www.ncl.org.br/ferramentas/composer_v2.1.0_win_setup.exe
- ✓ Máquina Virtual Java.
 - Pode-se obtê-la aqui: <http://java.sun.com/>
 - As versões atuais em Java não oferecem suporte a *alfa blending*, transparência, efeitos de transição, entre outros.
- ✓ Tutorial Ginga NCL 3.0
 - Disponível para download em:
 - <http://www.ncl.org.br/documentos/TutorialNCL3.0-2ed.pdf>

Anexos: Conectores

Conectores pré-definidos na ferramenta Composer

Modelo de conector simples

```
<causalConnector id="condiçãoAção">
  <simpleCondition role="condição" />
  <simpleAction role="ação" />
</causalConnector>
```

Exemplo:

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

"Quando a mídia associada ao papel **onBegin** for iniciada, inicie a apresentação associada ao papel **start**."

Os conectores pré-definidos que seguem esse modelo podem ser vistos na tabela:

ação \ condição	Start	Stop	Pause	Resume	Set + parâmetro var
onBegin	onBeginStart	onBeginStop	onBeginPause	onBeginResume	onBeginSet
onEnd	onEndStart	onEndStop	onEndPause	onEndResume	onEndSet
onSelection (seleção por mouse)	onSelectionStart	onSelectionStop	onSelectionPause	onSelectionResume	onSelectionSet
onSelection + parâmetro key (seleção por tecla)	onKeySelection-Start	onKeySelection-Stop	onKeySelection-Pause	onKeySelection-Resume	onKeySelection-Set

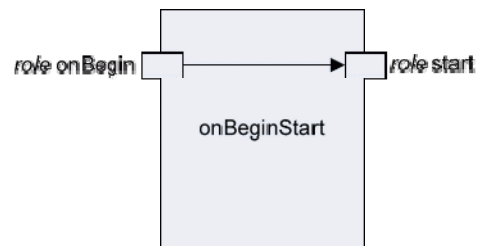
Conector onBeginStart

Nome: **onBeginStart**

Condição: início de exibição de um nó de mídia (mapeado no papel *onBegin*)

Ação: dispara o início de exibição de um nó de mídia (mapeado no papel *start*)

Ilustração:



Código NCL:

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

Leitura: Quando **<onBegin>** for iniciado, inicia **<start>**
(Substituí o papel entre **< >** pelo nó de mídia mapeado ao papel, no elo)

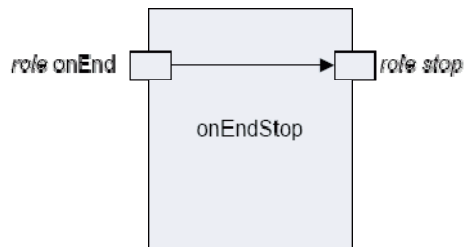
Conector onEndStop

Nome: onEndStop

Condição: término da exibição de um nó de mídia (mapeado no papel *onEnd*)

Ação: encerra a exibição de um nó de mídia (mapeado no papel *stop*)

Ilustração:



Código NCL:

```
<causalConnector id= "onEndStop">  
  <simpleCondition role= "onEnd" />  
  <simpleCondition role= "stop" />  
</causalConnector>
```

Leitura: Quando <onEnd> terminar, termina <stop>
(Substitui o papel entre < > pelo nó de mídia mapeado ao papel, no elo)

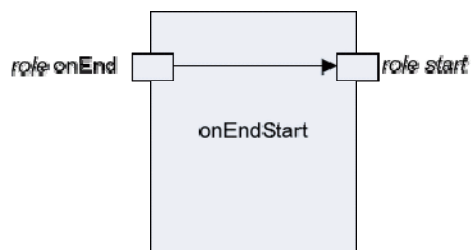
Conector onEndStart

Nome: onEndStart

Condição: término da exibição de um nó de mídia (mapeado no papel *onEnd*)

Ação: dispara o início da exibição de um nó de mídia (mapeado no papel *start*)

Ilustração:



Código NCL:

```
<causalConnector id="onEndStart">  
  <simpleCondition role="onEnd" />  
  <simpleCondition role="start" />  
</causalConnector>
```

Leitura: Quando <onEnd> terminar, inicia <start>
(Substitui o papel entre < > pelo nó de mídia mapeado ao papel, no elo)

Conector onBeginSet

No caso de uma ação de atribuição (papel set), os conectores definem um parâmetro var para receber o valor mapeado no elo, como a seguir:

Conector com ação de atribuição

```
<causalConnector id="onBeginSet">  
  <connectorParam name="var"/>  
  <simpleCondition role="onBegin"/>  
  <simpleAction role="set" value="$var"/>  
</causalConnector>
```

Exemplo:

```
<link id="IniciarVideo" xconnector=  
  "connBase#onBeginSetN">  
  <bind component="video1" role="onBegin"/>  
  <bind component="video2" interface="visible"  
    role="set">  
    <bindParam name="var" value="false"/>  
  </bind>  
</link>
```

“Quando a apresentação de video1 for iniciada, defina o valor da propriedade visible do nó video2 como false.”

Conector onKeySelection

No caso de uma seleção por tecla do controle remoto, define-se um parâmetro keyCode que indica qual tecla foi pressionada, como a seguir:

Conector com condição de acionamento de tecla

```
<causalConnector  
id="onKeySelectionStart">  
  <connectorParam name="keyCode"/>  
  <simpleCondition role="onSelection"  
key="$keyCode"/>  
  <simpleAction role="start"/>  
</causalConnector>
```

Exemplo:

```
<link id="RedStart" xconnector="connBase#  
onKeySelectionStart">  
  <bind component="video1" role="onSelection"/>  
  <bindParam name="keyCode" value="RED"/>  
  <bind component="video2" interface="visible"  
role="set">  
  </bind>  
</link>
```

“Quando a apresentação de video1 for iniciada, defina o valor da propriedade visible do nó video2 como false.”

Conectores predefinidos que permitem múltiplos mapeamentos para um único papel

Modelo de conector com cardinalidade N

```
<causalConnector id="condiçãoAção" >
  <simpleCondition role="condição" />
  <simpleAction role="ação"
  max="unbounded" qualifier="par" />
</causalConnector>
```

- **unbounded**: não há limite para o n.º de mapeamentos ao papel "ação".
- **par**: todas as associações serão consideradas em paralelo.

Exemplo:

```
<causalConnector id="onBeginStartN" >
  <simpleCondition role="onBegin" />
  <simpleAction role="start" max="unbounded"
  qualifier="par" />
</causalConnector>
```

"Quando a mídia associada ao papel **onBegin** for iniciada, **inicie ao mesmo tempo**, a apresentação de **todas as mídias** associada ao papel **start**."

Para definir um elo que utiliza esse tipo de conector, basta criar múltiplos mapeamentos (elementos do tipo **bind**) para cada papel:

```
<link id="BeginVideo1_img1_img2" xconnector="connBase# onBeginStartN">
  <bind component="video1" role="onBegin" />
  <bind component="imagem1" role="start">
  <bind component="imagem2" role="start">
</link>
```

Conectores para realizar ações de diferentes tipos ao mesmo tempo

Modelo de conector com ações compostas

```
<causalConnector id="condiçãoAção" >
  <simpleCondition role="condição" />
  <compoundAction operator="par">
    <simpleAction role="ação1"
  max="unbounded" qualifier="par" />
    <simpleAction role="ação2"
  max="unbounded" qualifier="par" />
  </compoundAction>
</causalConnector>
```

Exemplo:

```
<causalConnector id="onBeginStartNStopN" >
  <simpleCondition role="onBegin" />
  <compoundCondition operator="par">
    <simpleAction role="start" max="unbounded"
  qualifier="par" />
    <simpleAction role="stop" max="unbounded"
  qualifier="par" />
  </compoundCondition>
</causalConnector>
```

"Quando a mídia associada ao papel **onBegin** for iniciada, **ao mesmo tempo**, inicie a apresentação de **todas as mídias** associada ao papel **start** e termine a apresentação de **todas as mídias** associadas ao papel **stop**."

Conector onKeySelectionStartNStopNAbortN

Nome: onKeySelectionStartNStopNAbortN

- Condição:** tecla <keyCode> acionada (papel *onSelection*)
- Ação:**
= exibe as mídias identificadas pelo papel *start*
= pára as mídias identificadas pelo papel *stop*
= aborta as mídias identificadas pelo papel *abort*
- Código NCL:**

```
<causalConnector id="onKeySelectionStartNStopNAbortN">
  <connectorParam name="keyCode"/>
  <simpleCondition role="onSelection" key="$keyCode"/>
  <compoundAction operator="par">
    <simpleAction role="start" max="unbounded" qualifier="par" />
    <simpleAction role="stop" max="unbounded" qualifier="par" />
    <simpleAction role="abort" max="unbounded" qualifier="par" />
  </compoundAction>
</causalConnector>
```
- Leitura:** Quando a tecla <keyCode> for selecionada, inicia as mídias <start>, pára as mídias <stop> e aborta as mídias <abort>
- Observação:** Os elos que utilizarem este conector deverão identificar, como parâmetro adicional da ligação com o nó de origem (*bindParam* ou *linkParam*), o código virtual da tela do controle remoto associada à seleção através do parâmetro *keyCode*. Exemplo:

```
<bindParam name="keyCode" value="GREEN" />
```

Os códigos de teclas definidos no formatador atual são:

= RED, GREEN, YELLOW, BLUE

= VK_LEFT, VK_RIGHT, VK_UP, VK_DOWN

= VK_ENTER

= 1,2,3,4,5,6,7,8,9

= MENU

= e outros dependentes do dispositivo de camada: INTERACTION, *, letras A-Z.

Conector onBeginSetStartN

Nome:	onBeginSetStartN
Condição:	início de exibição da(s) mídia(s) no papel <i>onBegin</i>
Ação:	= altera propriedades dos nós associados ao papel <i>set</i> , conforme os valores passados pelo mapeamento (bindParam) para o parâmetro do conector (connectorParam) denominado <i>var</i> . = pára as mídias identificadas pelo papel <i>stop</i> = inicia a apresentação da(s) mídia(s) no papel <i>start</i> .
Código NCL:	<pre><causalConnector id="onBeginSetStartN"> <connectorParam name="var"/> <simpleCondition role="onBegin"/> <compoundAction operator="seq"> <simpleAction role="set" value="\$var" /> <simpleAction role="start" max="unbounded" qualifier="par" /> </compoundAction> </causalConnector></pre>
Observação:	Os elos que utilizarem este conector deverão identificar, como parâmetro adicional do mapeamento do nó (bindParam), as novas coordenadas e dimensões (x,y,w,h). Exemplo: <pre><bind component="video1" interface="bounds" interface="set" /> <bindParam name="var" value="0,0,100%,100%" /> </bind></pre>

Conector onEndSetStopN

Nome:	onEndSetStopN
Condição:	término de exibição da mídia no papel <i>onEnd</i>
Ação:	= altera propriedades dos nós associados ao papel <i>set</i> , conforme os valores passados pelo mapeamento (bindParam) para o parâmetro do conector (connectorParam) denominado <i>var</i> . = termina a apresentação das mídias identificadas pelo papel <i>stop</i>
Código NCL:	<pre><causalConnector id="onEndSetStopN"> <connectorParam name="bounds"/> <simpleCondition role="onEnd"/> <compoundAction operator="seq"> <simpleAction role="set" value="\$bounds" /> <simpleAction role="stop" max="unbounded" qualifier="par" /> </compoundAction> </causalConnector></pre>
Observação:	Como no caso anterior, os elos que utilizarem este conector deverão identificar, como parâmetro adicional do mapeamento do nó (bindParam), as novas coordenadas e dimensões (x,y,w,h).

Conector onKeySelectionSetN

Nome: onKeySelectionSetN

Condição: tecla <keyCode> acionada (papel *onSelection*)

Ação: = altera propriedades dos nós associados ao papel *set*, conforme os valores passados pelo mapeamento (**bindParam**) para o parâmetro do conector (**connectorParam**) denominado *var*.

Código NCL:

```
<causalConnector id="onKeySelectionSetN">
  <connectorParam name="keyCode"/>
  <connectorParam name="var"/>
  <simpleCondition role="onSelection" key= "$keyCode" > />
  <simpleAction role= "set" value="$var" max="unbounded" qualifier="par" />
</causalConnector>
```

Observação: Os elos que utilizarem este conector deverão identificar, como parâmetro adicional do elo (**linkParam**), ou como novos parâmetros do mapeamento de um nó (**bindParam**), o código virtual da tecla do controle remoto associada à seleção.
Exemplo:

```
<bindParam name= "keyCode" value="VK_ENTER" />
```

Os elos que utilizarem este conector deverá identificar também, como parâmetro adicional dos mapeamentos dos nós (**bindParam**), os novos valores das propriedades. No exemplo, as propriedades de visibilidade e volume do som, como a seguir:

```
<bind component="video1" interface="visible" role="set">
  <bindParam name="var" value="false"/>
</bind>
<bind component="video1" interface="soundLevel" role="set">
  <bindParam name="var" value="0"/>
</bind>
```

Introduzir retardos na realização de uma ação:

```
<causalConnector id="onBeginStartNDelayStopN">
  <connectorParam name="delay" />
  <simpleCondition role="onBegin" />
  <compoundAction operator="seq">
    <simpleAction role="start" max="unbounded" qualifier="par" />
    <simpleAction role="stop" delay="$delay" max="unbounded" qualifier="par" />
  </compoundAction>
</causalConnector>
```

Um conector pode definir também condições compostas. Neste caso, o operador (operator) define se todas as condições precisam ser satisfeitas para a ativação do elo (operator="and") ou se o elo é ativado quando qualquer uma das condições definidas é satisfeita (operator="or"). Com frequência, definem-se os conectores que testam o valor de uma determinada propriedade para decidir sobre a ativação do elo. No exemplo a seguir, quando a mídia associada ao papel **onBegin** é iniciada, verifica-se se o valor da propriedade do nó mapeada ao papel **attNodeTest** é igual a **value** e, caso seja, o elo é ativado.

```
<causalConnector id="onBeginAttNodeTestStartN">
  <connectorParam name="value" />
  <compoundAction operator="and">
    <simpleCondition role="onBegin" />
    <assessmentStatement comparator="eq">
      <attributeAssessment role="attNodeTest" eventType="attribution"
        attributeType="nodeAttribute" />
      <valueAssessment value="$value" />
    </assessmentStatement>
    <simpleAction role="start" max="unbounded" qualifier="par" />
  </compoundAction>
</causalConnector>
```

Regras:

As regras simples de um documento NCL são definidas na seção `<ruleBase>`, com base numa propriedade, operador e valor, como no exemplo a seguir:

```
<rulerBase>
  <rule id="rEn" var="idioma" comparator="eq" value="en" />
  <rule id="rPt" var="idioma" comparator="eq" value="pt" />
</rulerBase>
```

Os seguintes operadores podem ser utilizados como comparador na definição de regras:

- `eq` (*equal* - igual a)
- `ne` (*not equal* - diferente de);
- `gt` (*greater than* - maior que);
- `ge` (*greater than or equal to* - maior ou igual a);
- `lt` (*less than* - menor que);
- `le` (*less than or equal to* - menor ou igual a).

Uma forma de armazenar as propriedades que serão utilizadas nas regras é utilizar o nó `settings`. Trata-se de um nó de propriedades globais, como no seguinte exemplo:

```
<media type="application/x-ginga-settings" id="nodeSettings">
  <property name="idioma" />
</media>
```

Switch:

Um **switch** é um contexto com nós alternativos, ou seja, dentro os quais apenas um será ativado. A decisão sobre qual nó será ativado é dada por regras. Na definição de um **switch**, além dos nós que o compõem, devem ser definidos os mapeamentos das regras para esses nós e suas âncoras, tal como no seguinte exemplo:

```
<switch id="switchAudioldioma" >
  <bindRule rule="rEn" constituent="audioEn"/>
  <bindRule rule="rPt" constituent="audioPt"/>

  <media type="audio" id="audioEn" src="media/audioEn.mp3" descriptor="dAudio1"/>
  <media type="audio" id="audioPt" src="media/audioPt.mp3" descriptor="dAudio2"/>
</switch>
```

Caso o **switch** seja composto de contextos, é necessário ainda definir um ou mais elementos **switchPort** para indicar a porta de destino de cada mapeamento das regras, como no seguinte exemplo:

```
<switch id="switchNivel" >
  <switchPort id="pNivel" >
    <mapping component="ctxBasico" interface="pBasico" />
    <mapping component="ctxAvancado" interface="pAvancado" />
  </switchPort>
  <bindRule rule="rBasico" component="ctxBasico" />
  <bindRule rule="rAvancado" component="ctxAvancado" />

  <context id="ctxBasico" >
    <port id="pBasico" component="videoBasico" />
    <!-- nós e elos do contexto ctxBasico -->
  </context>
  <context id="ctxAvancado" >
    <port id="pAvancado" component="videoAvancado" />
    <!-- nós e elos do contexto ctxAvancado-->
  </context>
</switch>
```

Conector onKeySelectionSetNStartNStopNAbortN

Nome: onKeySelectionSetNStartNStopNAbortN

Condição: tecla <keyCode> acionada (papel *onSelection*)

Ação:
= exibe as mídias identificadas pelo papel *start*
= pára as mídias identificadas pelo papel *stop*
= aborta as mídias identificadas pelo papel *abort*; e
= define o valor <var> à propriedade mapeada através do papel *set*

Código NCL:

```
<causalConnector id="onSelection1SetNStartNStopNAbortN" >  
  <connectorParam name="keyCode" />  
  <connectorParam name="var" />  
  <simpleCondition role="onSelection" key="$keyCode" />  
  <compoundAction operator="seq" >  
    <simpleAction role="set" value="$var" max="unbounded" qualifier="par" />  
    <simpleAction role="start" max="unbounded" qualifier="par" />  
    <simpleAction role="stop" max="unbounded" qualifier="par" />  
    <simpleAction role="abort" max="unbounded" qualifier="par" />  
  </compoundAction>  
</causalConnector>
```

Observação: Os elos que utilizarem este conector deverão identificar, como parâmetro adicional do elo (**linkParam**), ou como novos parâmetros do mapeamento de um nó (**bindParam**), o código virtual da tecla do controle remoto associada à seleção. Exemplo:

```
<linkParam name="keyCode" value="VK_ENTER" />
```

Os elos que utilizarem este conector deverá identificar também, como parâmetro adicional dos mapeamentos dos nós (**bindParam**), os novos valores das propriedades. Por exemplo:

```
<bindParam name="var" value="en" />
```

Conector onKeySelectionSetNStopNDStartNDStopNDAbortN

Nome: onKeySelectionSetNStopNDStartNDStopNDAbortN

Condição: tecla <keyCode> acionada (papel *onSelection*)

Ação:
= pára as mídias identificadas pelo papel *stop*
= define o valor <var> à propriedade mapeada através do papel *set*
= exhibe, após um *delay* de 0.5s, as mídias identificadas pelo papel *dstart*
= pára, após um *delay* de 0.5s, as mídias identificadas pelo papel *dstop*
= aborta, após um *delay* de 0.5s, as mídias identificadas pelo papel *dabort*

Código NCL:

```
<causalConnector id="onKeySelectionSetNStopNDStartNDStopNDAbortN">
  <connectorParam name="keyCode" />
  <connectorParam name="var" />
  <simpleCondition role="onSelection" key="$keyCode" />
  <compoundAction operator="par" >
    <simpleAction role="set" value="$var" max="unbounded" qualifier="par" />
    <simpleAction role="stop" max="unbounded" qualifier="par" />
    <simpleAction role="dstart" delay="0.5s" max="unbounded" qualifier="par" />
    <simpleAction role="dstop" delay="0.5s" max="unbounded" qualifier="par" />
    <simpleAction role="dabort" delay="0.5s" max="unbounded" qualifier="par" />
  </compoundAction>
</causalConnector>
```

Observação: Os elos que utilizarem este conector deverão identificar, como parâmetros adicionais da ligação com o nó de origem (*bindParam* ou *linkParam*), o código virtual da tecla do controle remoto associada à seleção e o valor a ser atribuído. Exemplo:

```
<bindParam name="keyCode" value="GREEN" />
<bindParam name="var" value="en" />
```

Fazendo uso deste conector, pode-se ocultar imediatamente a opção não selecionada, para dar um **feedback** ao usuário sobre qual opção ele escolheu e, após 0,5 seg. oculta-se a opção selecionada e substitui-se o *video1* pelo *video2*.

Conector onSelectionSetNStartStopN

Nome: onSelectionSetNStartStopN

Condição: mídia selecionada, ou seja, mídia com o foco quando a tecla OK é pressionada (papel *onSelection*)

Ação:
= define o valor <var> às propriedades mapeadas através do papel *set*
= inicia as mídias identificadas pelo papel *start*
= pára as mídias identificadas pelo papel *stop*

Código NCL:

```
<causalConnector id="onSelectionSetNStartNStopN" >  
  <connectorParam name="var" />  
  <simpleCondition role="onSelection" />  
  <compoundAction operator="seq" >  
    <simpleAction role="set" value="$var" max="unbounded" qualifier="par" />  
    <simpleAction role="stop" max="unbounded" qualifier="par" />  
    <simpleAction role="start" max="unbounded" qualifier="par" />  
  </compoundAction>  
</causalConnector>
```

Observação: Os elos que utilizarem este conector deverão identificar, como parâmetro adicional dos mapeamentos dos nós (**bindParam**), os valores das propriedades a serem atribuídas (var). Exemplo:

```
<bind component="nodeSettings" interface="opcao" role="set" >  
  <bindParam name="var" value="3" />  
</bind>
```