



PUC
RIO



Construindo Programas Audiovisuais Interativos Utilizando a NCL 3.0 e a Ferramenta Composer

Carlos de Salles Soares Neto

Luiz Fernando Gomes Soares

Rogério Ferreira Rodrigues

Simone Diniz Junqueira Barbosa

2^a edição: 31/07/2007

Sumário

Introdução	11
Estrutura de um documento hipermídia	12
O que tocar?	12
Onde tocar?	13
Como tocar?	14
Quando tocar? (parte I)	15
Estrutura de um documento NCL.....	16
A ferramenta Composer	18
Seu primeiro documento NCL.....	21
Exemplo 01 – Reprodução de um objeto de mídia.....	21
Visões do documento.....	21
Visão estrutural.....	21
Visão de leiaute	21
Visões temporal e espacial	22
Passo-a-passo no Composer.....	22
Passo 0: iniciando um novo projeto no Composer	23
Passos 1 e 2: definindo regiões de tela	24
Passo 3: definindo o descritor que determina como o vídeo será exibido	30
Passo 4: definindo a mídia propriamente dita.....	35
Passo 5: definindo a porta do contexto <i>body</i> que determina “onde” o programa inicia (apresentando qual mídia)	39
Passo 6: Testando o documento.....	41
Sincronizando nós de mídia através de elos e conectores.....	46
Exemplo 02 – Iniciando e terminando dois objetos de mídia simultaneamente.....	46
Visões do documento.....	46
Visão Estrutural	46
Visão de leiaute	46
Visões Temporal e Espacial	47
Passo-a-passo no Composer.....	47
Passo 1: criando região para o título.....	48
Passo 2: criando descritor que determina como o título será exibido.....	49
Passo 3: definindo a mídia correspondente ao título	49
Passos 4 e 5: criando elos de sincronismo	50
Exemplo 03 – Iniciando um objeto de mídia quando outro termina	58
Visões do documento.....	59
Visão estrutural.....	59

Visão de leiaute	59
Visões temporal e espacial	59
Passo-a-passo no Composer.....	59
Passo 1: definindo a mídia correspondente ao segundo vídeo.....	60
Passo 2: criando um elo de sincronismo.....	60
Sincronizando diversos nós de mídia estática a um nó de mídia contínua	63
Exemplo 04 – Sincronizando um vídeo com diferentes arquivos de legenda	63
Visões do documento.....	63
Visão estrutural.....	63
Visão de leiaute	64
Visões temporal e espacial	64
Passo-a-passo no Composer.....	64
Passo 1: criando região para a legenda.....	65
Passo 2: criando descritor para a legenda.....	65
Passo 3: definido nós de texto HTML correspondentes às legendas.....	66
Passo 4: definindo âncoras no vídeo.....	67
Passo 5: criando elos de sincronismo para as legendas	68
Exemplo 05 – Sincronizando um vídeo com um único arquivo de legenda, segmentado	71
Interação com o Usuário	73
Exemplo 06 – Exibindo um vídeo em loop até a intervenção do usuário	73
Visões do documento	73
Visão estrutural	73
Visão de leiaute.....	74
Visões temporal e espacial	74
Alternativa a: novo arquivo de conectores	74
Alternativa b: definição de novo conector embutida no documento NCL	75
Redimensionando regiões.....	77
Exemplo 07 – Redimensionando uma região de vídeo durante a exibição de uma imagem... 77	77
Visões do documento.....	77
Visão estrutural.....	77
Visão de leiaute	77
Visões temporal e espacial	78
Exemplo 08 – Trocando um objeto de mídia em resposta a uma ação do usuário	80
Visões do Documento	81
Visão estrutural.....	81
Visão de leiaute	81
Visões temporal e espacial	82
Exemplo 09 – Alternando imagens para identificar ações disponíveis	85

Visões do documento	85
Visão estrutural.....	85
Visão de leiaute	86
Visões temporal e espacial	86
Adaptação do comportamento do programa	88
Exemplo 10 – Simulação de um menu de DVD.....	88
Visões do documento.....	88
Visão estrutural.....	88
Visão de leiaute	89
Visões Temporal e Espacial	89
Exemplo 11 – Simulação de um menu de DVD com feedback	92
Visões do documento	93
Visão estrutural	93
Visões temporal e espacial.....	93
Menu	95
Exemplo 12 – Seleção através das setas ou números do controle remoto.....	95
Re-uso de nós.....	98
Apêndice I – Listagens dos exemplos	100
Exemplo 01 – Seu primeiro documento NCL: reprodução de um objeto de mídia	100
Exemplo 02 – Iniciando e terminando dois objetos de mídia simultaneamente	102
Exemplo 03 – Iniciando um objeto de mídia quando outro termina.....	105
Exemplo 04 – Sincronizando um vídeo com diferentes arquivos de legenda.....	108
Exemplo 05 – Sincronizando um vídeo com um único arquivo de legenda, segmentado	111
Exemplo 06 – Exibindo um vídeo em loop até a intervenção do usuário	114
Exemplo 07 – Redimensionando uma região de vídeo durante a exibição de uma imagem.....	117
Exemplo 08 – Trocando um objeto de mídia em resposta a uma ação do usuário	120
Exemplo 09 – Alternando imagens para identificar ações disponíveis.....	124
Exemplo 10 – Simulação de um menu de DVD	127
Exemplo 11 – Simulação de um menu de DVD com feedback	132
Exemplo 12 – Seleção através das setas do controle remoto	111
Exemplo 13 – Reutilização de nós.....	141
Apêndice II – Conectores Predefinidos.....	149

Figuras

Figura 1. Nós e elos num hipertexto comum.....	11
Figura 2. Nós, elos e nós de composição (contextos).....	11
Figura 3. Representação de mídia utilizada neste documento.....	12
Figura 4. Representação de um nó de composição.....	13
Figura 5. Representação de região utilizada neste documento.....	13
Figura 6. Representação de um descritor associado a uma região.....	14
Figura 7. Descritores fazem a associação de uma mídia com uma região.....	15
Figura 8. Porta de um nó de composição.	15
Figura 9. Ferramenta de autoria Composer.	18
Figura 10. Visão estrutural da ferramenta Composer.....	19
Figura 11. Visão temporal da ferramenta Composer.....	19
Figura 12. Visão de leiaute da ferramenta Composer.	20
Figura 13. Visão textual da ferramenta Composer.	20
Figura 14. Visões estrutural e de leiaute do exemplo 01, para reprodução de um único vídeo, sem sincronismo ou interação com o usuário.	21
Figura 15. Visões temporal e espacial do exemplo 01, para reprodução de um único vídeo, sem sincronismo ou interação com o usuário.	22
Figura 16. Menu File do Composer.....	23
Figura 17. Quadro de diálogo para definição do nome e diretório do projeto.....	23
Figura 18. Menu de acesso à visão de leiaute.....	24
Figura 19. Áreas da visão de leiaute no Composer.	25
Figura 20. Possibilidades de acesso à criação de uma nova região.	25
Figura 21. Quadro de diálogo para definição dos atributos de uma região, a) com valores defaults; e b) redefinidos pelo usuário.	26
Figura 22. Região recém-criada na visão de leiaute.	26
Figura 23. Definição dos atributos da região rgVideo1.....	27
Figura 24. Visão de leiaute apresentando a região rgVideo1, criada dentro da rgTV.....	27
Figura 25. Atributos de posicionamento e dimensão de uma região.	29
Figura 26. Criação de um descritor para a região rgVideo1.....	30
Figura 27. Visão estrutural: a) acesso e b) visão estrutural vazia.....	35
Figura 28. Criando um nó de mídia: a) acesso via menu; b) quadro de diálogo dos atributos de um nó de mídia.....	36
Figura 29. Atributos do nó de vídeo do exemplo 01.	36
Figura 30. Quadro de diálogo para associação de um descritor ao nó de mídia.....	37
Figura 31. Visão estrutural exibindo o nó de mídia recém criado, video1.....	37

Figura 32. Definindo uma porta para o nó de mídia (a) e indicação visual do nó com porta no contexto body (b).....	40
Figura 33. Porta pInicio como ponto de entrada a um nó interno de um contexto.....	40
Figura 34. Visões estrutural e de leiaute do exemplo 02.....	46
Figura 35. Visões temporal e espacial do exemplo 02, com sincronismo através dos conectores onBeginStart e onEndStop	47
Figura 36. Região para exibir o título do vídeo video1.....	48
Figura 37. Visão de leiaute após a criação da região rgTitulo1.....	49
Figura 38. Quadro de diálogo para definição do descritor dTitulo1.....	49
Figura 39. Quadro de diálogo para criação de nó de texto titulo1.....	50
Figura 40. Visão estrutural após a criação do nó titulo1.....	50
Figura 41.Visão estrutural com dois nós selecionados.....	51
Figura 42. Item de menu Insert > Causal link.....	51
Figura 43. Quadro de diálogo para criação de um elo.....	52
Figura 44. Quadros de diálogo para criação do elo lBeginVideo1StartTitulo1.....	53
Figura 45. Visão estrutural exibindo elo lBeginVideo1Titulo1 entre os nós video1 e titulo1.....	53
Figura 46. Quadro de diálogo para a criação do elo lEndVideo1StopTitulo1.....	54
Figura 47. Visão estrutural apresentando os elos lBeginVideo1StartTitulo1 e lEndVideo1StopTitulo1.....	54
Figura 48. Ilustração de um conector ligando três nós.....	56
Figura 49. Máquina de estados de eventos.....	57
Figura 50. Visões estrutural e de leiaute do exemplo 03, onde o término de um vídeo dispara outro.....	59
Figura 51. Visões temporal e espacial do exemplo 03.....	59
Figura 52. Criação do nó video2.....	60
Figura 53. Visão estrutural apresentando os nós video1 e video2.....	60
Figura 54. Criação do elo lEndVideo1StartVideo2.....	61
Figura 55. Visão estrutural apresentando o elo lEndVideo1StartVideo2.....	61
Figura 56. Visões estrutural e de leiaute do exemplo 04, de sincronização de legendas a segmentos de um vídeo.....	63
Figura 57. Visões temporal e espacial do exemplo 04.....	64
Figura 58. Criando região para a legenda.....	65
Figura 59. Criando descritor para a legenda.....	66
Figura 60. Criando um nó de texto para a legenda.....	66
Figura 61. Visão estrutural apresentando um nó de vídeo e três nós HTML correspondentes a legendas.....	67
Figura 62. Criando âncora para a primeira legenda.....	67
Figura 63. Nó video1 com as três âncoras definidas.....	68

Figura 64. Criação de um elo que utiliza uma âncora em um de seus papéis.	69
Figura 65. Visão temporal ilustrando o sincronismo entre âncoras de vídeo e as respectivas legendas.	69
Figura 66. Criação de nó com fragmento de arquivo.	72
Figura 67. Visão estrutural do exemplo 06, com sincronismo (conectores onBeginStart e onEndStart) e interatividade (conector onKeySelectionStartNStopNAbrtN).	73
Figura 68. Visão de leiaute do exemplo 06.	74
Figura 69. Visões temporal e espacial do exemplo 06.	74
Figura 70. Visão estrutural do exemplo 07, documento que redimensiona um vídeo enquanto uma imagem é exibida.	77
Figura 71. Visão de leiaute do exemplo 07.	77
Figura 72. Visões temporal e espacial do exemplo 07.	78
Figura 73. Visão estrutural do exemplo 08.	81
Figura 74. Visão de leiaute do exemplo 08.	81
Figura 75. Visões temporal e espacial do exemplo 08.	82
Figura 76. Visão estrutural do exemplo 09.	85
Figura 77. Visão de leiaute do exemplo 09.	86
Figura 78. Visões temporal e espacial do exemplo 09.	86
Figura 79. Visão estrutural do exemplo 10.	88
Figura 80. Visão de leiaute do exemplo 10.	89
Figura 81. Visões temporal e espacial do exemplo 10.	89
Figura 82. Visão estrutural do exemplo 11.	93
Figura 83. Visões temporal e espacial do exemplo 11.	93
Figura 84. Esquema de seleção de uma dentre 6 opções de menu.	95
Figura 85. Indicação da opção atual selecionada.	96
Figura 86. Visão estrutural parcial de um nó videoPrincipal sendo reutilizado em dois outros contextos.	98
Figura 87. Situações inválidas de re-uso de nó.	99

Tabelas

Tabela 1. Estrutura básica de um documento NCL.....	17
Tabela 2. Parâmetros que podem ser utilizados em descriptorParam, conforme a mídia.	33
Tabela 3. Definição do conector onBeginStart.....	57
Tabela 4. Definição do conector onEndStop.....	58
Tabela 5. Definição do conector onEndStart.....	62
Tabela 6. Conector onKeySelectionStartNStopNAbortN.	76
Tabela 7. Conector onBeginSetStartN.	79
Tabela 8. Conector onEndSetStopN.	79
Tabela 9. Conector onKeySelectionSetN.	84
Tabela 10. Definição do conector onKeySelectionSetNStartNStopNAbortN.	92
Tabela 11. Redefinição do conector onKeySelectionSetNStopNDStartNDStopNDAbortN, para introduzir papéis de ativação com retardo.....	94
Tabela 12. Definição do conector onKeySelectionSetNStartNStopN, para atribuir o valor da opção selecionada na variável “opcao”, parar os botões e exibir o nó de alternativa.	96

Listagens

Listagem 1. Documento NCL para reprodução de um só vídeo.....	100
Listagem 2. Documento NCL com elos para sincronizar o início e o término de exibição de mídias.	102
Listagem 3. Documento NCL com elos para sincronizar o início de exibição de uma mídia quando outra termina.	105
Listagem 4. Documento NCL para reprodução de um vídeo com três trechos de legenda sincronizados.	108
Listagem 5. Documento NCL para reprodução de um vídeo com três trechos de legenda sincronizados, num único arquivo.	111
Listagem 6. Documento NCL para exibir um vídeo em loop e substituí-lo por outro quando o usuário pressionar a tecla verde.	114
Listagem 7. Documento NCL para redimensionamento de um vídeo enquanto uma imagem está sendo exibida.	117
Listagem 8. Documento NCL com elos para troca de nós de vídeo através da interação do usuário.	120
Listagem 9. Documento NCL com elos para alternar a exibição de imagens através da interação do usuário para refletir a alternância entre nós de vídeo e as oportunidades de interação.	124
Listagem 10. Documento NCL para exibir um vídeo em loop e, mediante a seleção do usuário, substituí-lo por um segundo vídeo e, finalmente, exibir um terceiro vídeo e um áudio conforme a seleção de idioma.	128
Listagem 11. Modificações sobre a listagem anterior (linhas 183 a 209) para, mediante a seleção do usuário, fornecer feedback momentâneo.....	132
Listagem 12.Documento NCL ilustrando um menu com 6 opções.....	136
Listagem 13. Documento NCL ilustrando re-uso num programa com dois modos de exibição — básico e avançado.....	141

Introdução

Este documento consiste num tutorial sobre a linguagem NCL (*Nested Context Language*), versão 3.0. Ele apresenta diversos exemplos de elaboração de documentos hipermídia, com sincronismo entre mídias e interação com o usuário.

Documentos hipermídia geralmente são compostos de nós (*nodes*) e elos (*links*) (Figura 1).

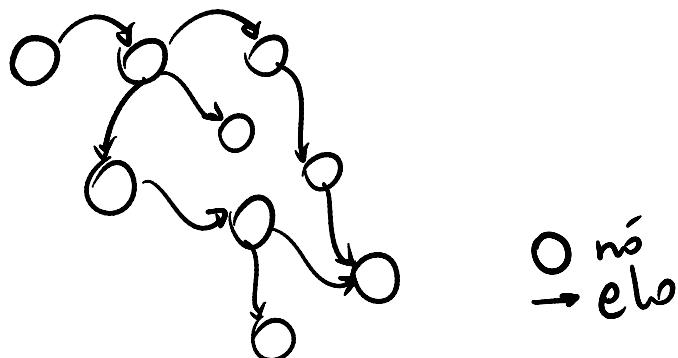


Figura 1. Nós e elos num documento hipermídia comum.

O NCM (*Nested Context Model*, Modelo de Contextos Aninhados), modelo subjacente à linguagem NCL, vai além dos grafos acima para representar hipermídia. No NCM, os grafos podem ser aninhados, permitindo segmentar e estruturar o documento hipermídia conforme necessário ou desejado. Isto é feito através de nós de composição, que são nós compostos de grafos NCM (Figura 2). Um nó de composição também é chamado de contexto.

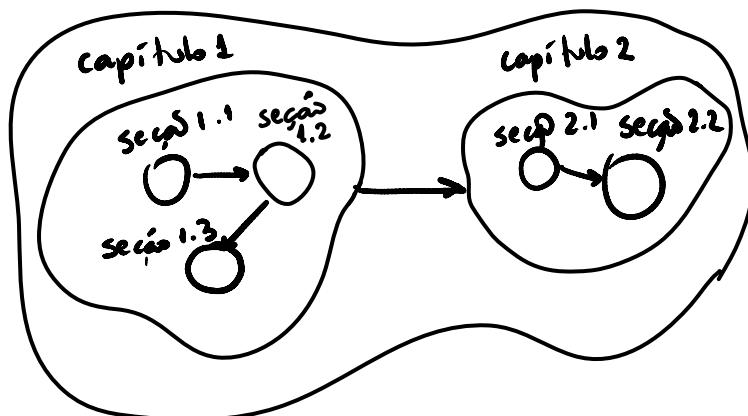


Figura 2. Nós, elos e nós de composição (contextos).

Assim, em NCM, um *node* (nó) pode ser de dois tipos:

- nó de conteúdo ou de mídia (*content node* ou *media node*): associado a um elemento de mídia como vídeo, áudio, imagem, texto, aplicação etc.; ou
- nó de composição ou contexto (*composite node* ou *context*, do qual *switch* é um caso particular, como será visto adiante).

No caso da Figura 2, capítulo 1 e capítulo 2 são contextos (nós de composição), enquanto cada seção é um nó de conteúdo.

Para auxiliar a construção de documentos hipermídia seguindo o modelo NCM, foi desenvolvida a linguagem NCL, que é utilizada por todo este documento na elaboração de exemplos de documentos hipermídia. Os elementos da linguagem serão introduzidos progressivamente, com base em exemplos. Os exemplos são baseados na versão atual do formatador (disponível no site www.ncl.org.br na mesma data deste documento), que é o programa que interpreta um documento NCL e apresenta o programa audiovisual interativo nele representado¹. Este documento apresenta também alguns conceitos do NCM versão 3.0 que julgamos necessários para o entendimento da NCL aqui apresentada.

Estrutura de um documento hipermídia

Para construir um documento hipermídia, é necessário definir **o que** se quer tocar, **onde** (i.e. em que região da tela de qual dispositivo), **como** (i.e. em que volume, com ou sem borda, com que *player*), e **quando** (antes/depois de qual mídia ser apresentada ou após qual tecla ser pressionada).

O que tocar?

Em geral, a primeira coisa que consideramos quando começamos a conceber um programa audiovisual interativo é o seu conteúdo. Esse conteúdo é representado através de **nós de mídia**. Neste documento, representamos um nó de mídia graficamente através de círculos, como ilustrado na Figura 3.

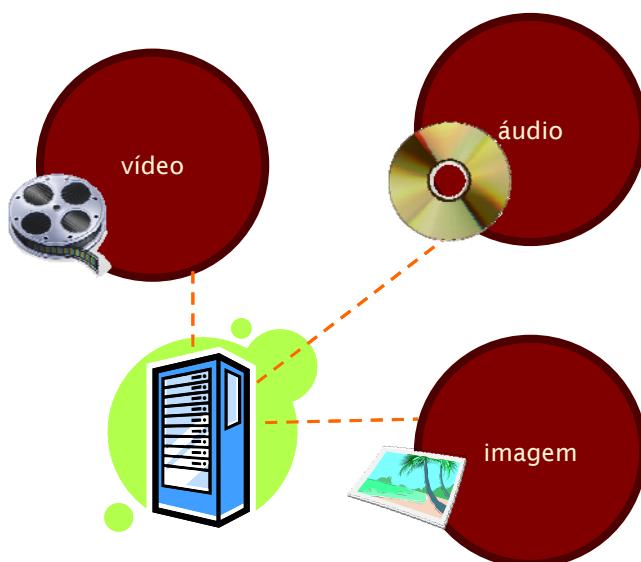


Figura 3. Representação de mídia utilizada neste documento.

¹ O formatador de programas escritos em NCL é como um programa do tipo *media player*, geralmente embarcado em um terminal de acesso ou em uma televisão, e com o qual o usuário pode interagir através de um controle remoto.

Todo nó de mídia é definido dentro de um contexto. Na NCL, o elemento **body** é o contexto que contém todos os nós do documento, sejam nós de mídia ou contextos. A Figura 4 ilustra um documento com 4 nós de mídia, 3 dos quais estão dentro de um contexto (*ctx1*) aninhado ao *body*.

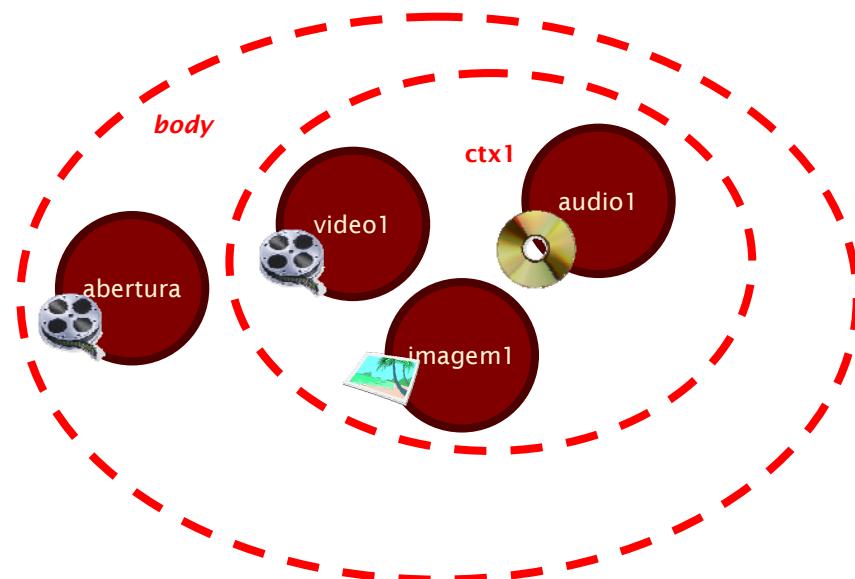


Figura 4. Representação de nós de mídia e de composição.

Onde tocar?

À medida que definimos o conteúdo do nosso programa, também começamos a definir as áreas onde cada mídia será apresentada na tela, através de elementos denominados **regiões**. Uma região indica a posição e as dimensões da área onde uma mídia será apresentada. Em outras palavras, uma região serve para inicializar a posição dos nós de mídia num local específico.

Neste documento, representamos uma região graficamente através de retângulos, como ilustrado na Figura 5.

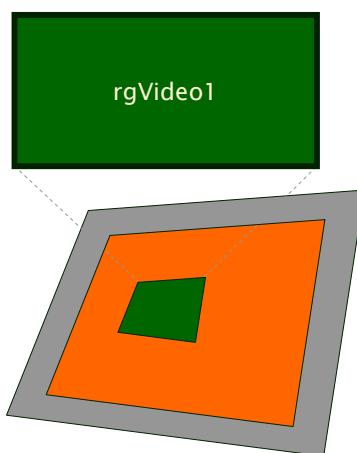


Figura 5. Representação de região utilizada neste documento.

É importante observar que uma região define onde mídias poderão ser apresentadas, mas não a associação com cada mídia particular. Essa associação é feita através de um descritor, como será visto na próxima seção.

Como tocar?

A associação de uma mídia a uma região é definida através de um **descritor**. Descritores são utilizados também para definir a forma como a mídia deverá ser apresentada. Por exemplo, um descritor de uma mídia de áudio pode definir o seu volume; o de uma mídia de imagem, algum grau de transparência; o de uma mídia de texto, se a mídia será apresentada visualmente ou lida por um sintetizador de voz, etc.

Ao se definir um descritor, é necessário definir a região à qual ele estará associado (Figura 6). Toda mídia que utilizar aquele descritor estará associada à região correspondente.

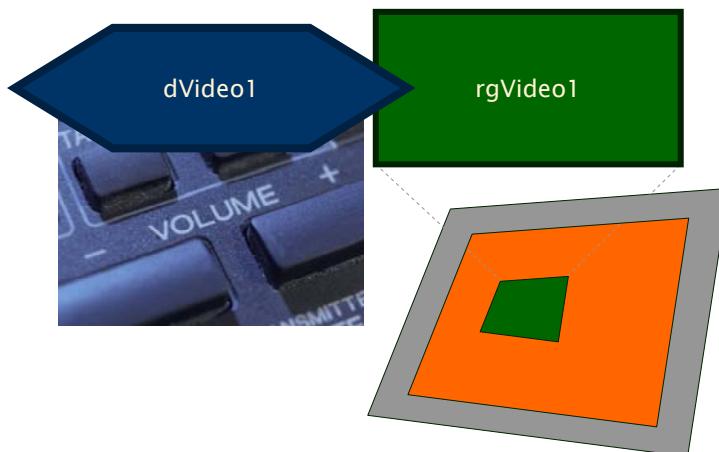


Figura 6. Representação de um descritor associado a uma região.

Mesmo que não se queira alterar a forma como uma mídia será apresentada, é necessário um descritor para **associar a mídia à região** onde deverá ser apresentada. Na Figura 7 ilustramos um descritor *dVideo1* utilizado por uma mídia *video1* que será apresentada na região *rgVideo1*, sem qualquer modificação na forma como será apresentada.

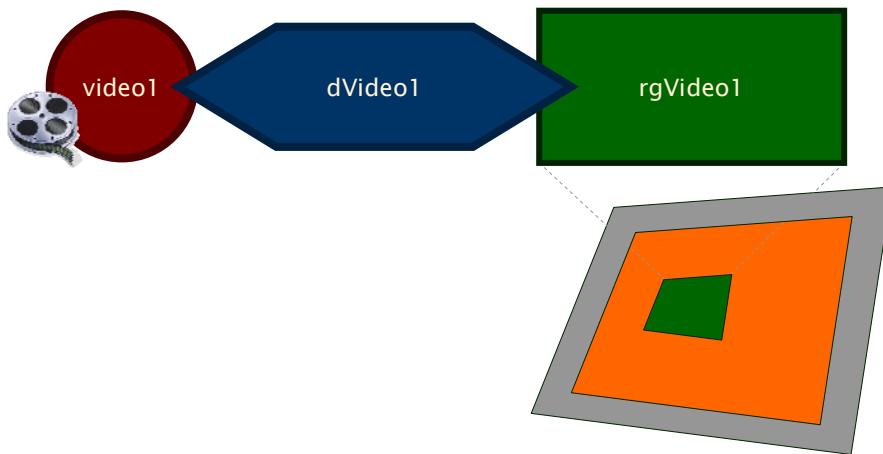


Figura 7. Descritores fazem a associação de uma mídia com uma região.

Quando tocar? (parte I)

Para definir qual é o **primeiro nó** do documento a ser apresentado, deve-se criar uma porta no contexto *body* para esse nó. Caso haja mais de uma porta no contexto *body*, os nós mapeados por todas as portas são iniciados em paralelo.

Identificar por onde o documento pode começar a ser apresentado é apenas uma característica específica de uma porta. De fato, portas são necessárias para dar acesso a nós (sejam nós de mídia ou contextos) internos a um contexto qualquer, e não apenas ao *body*. Na Figura 8, o nó *video1* do contexto *ctx1* só pode ser acessado, de fora do contexto *ctx1*, através da porta *pVideo1*. Já os nós *audio1* e *imagem1* não podem ser acessados de fora do contexto *ctx1*.

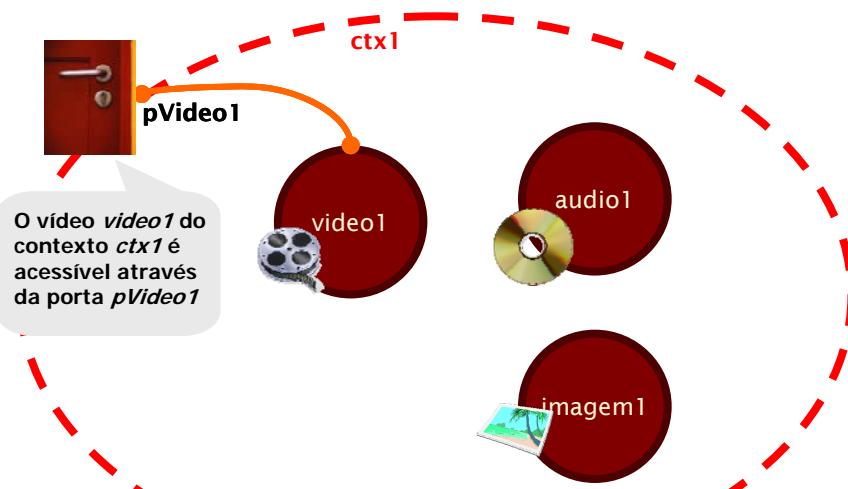


Figura 8. Porta de um nó de composição.

Para definir **quando** um nó de mídia será apresentado em relação a outros, são criados **elos**, que são utilizados para estabelecer o sincronismo entre os nós e para definir a interatividade

do programa. O comportamento desses elos é dado por **conectores**. Elos e conectores são vistos mais adiante, na seção “Quando Tocar? (parte II)”.

Estrutura de um documento NCL

Um documento NCL é um arquivo escrito em XML. Todo documento NCL possui a seguinte estrutura:

- um cabeçalho de arquivo NCL (linhas 1 e 2);
- uma seção de cabeçalho do programa (seção **head**, linhas 3 a 13), onde se definem as regiões, os descritores, os conectores e as regras utilizados 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 a estrutura do programa;
- pelo menos uma porta que indica por onde o programa começa a ser exibido (**port pInicio**, linha 15); e
- a conclusão do documento (linha 18).

A Tabela 1 apresenta a estrutura básica de um documento NCL.

Tabela 1. Estrutura básica de um documento NCL.

cabeçalho do arquivo NCL	1: <?xml version="1.0" encoding="ISO-8859-1"?> 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">	1
cabeçalho do programa	3: <head>	
base de regiões	4: <regionBase> 5: <!-- regiões da tela onde as mídias são apresentadas --> 6: </regionBase>	2
base de descritores	7: <descriptorBase> 8: <!-- descritores que definem como as mídias são apresentadas --> 9: </descriptorBase>	3
base de conectores	10: <connectorBase> 11: <!-- conectores que definem como os elos são ativados e o que eles disparam --> 12: </connectorBase>	8
	13: </head>	
corpo do programa	14: <body>	
ponto de entrada no programa	15: <port id="plnicio" component="ncPrincipal" interface="lInicio"/>	5
conteúdo do programa	16: <!-- contextos, nós de mídia e suas âncoras, elos e outros elementos -->	6 7
	17: </body>	
término	18: </ncl>	



Geralmente, os passos para se construir um documento NCL envolvem definir:

- ① os cabeçalhos básicos do arquivo NCL e do programa;
- ② as regiões da tela onde aparecerão os elementos visuais (**regionBase**);
- ③ como e onde os nós de mídia serão exibidos, através de descritores (**descriptorBase**);
- ④ o conteúdo (nós de mídia - **media**) e a estrutura (contextos - **context**) do documento (seção **body**), associando-os aos descritores;
- ⑤ 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**);
- ⑥ âncoras para os nós de mídia, visando à construção dos elos entre nós (**area** e **attribute**);
- ⑦ elos para o sincronismo e interatividade entre os nós de mídia e contextos (**link**); e
- ⑧ 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.

A ferramenta Composer

O Composer é uma ferramenta de autoria hipermídia desenvolvida pelo Laboratório TeleMídia² do Departamento de Informática da PUC-Rio. Com essa ferramenta, é possível construir programas audiovisuais interativos com pouco conhecimento da linguagem NCL. A Figura 9 apresenta uma tela do Composer:

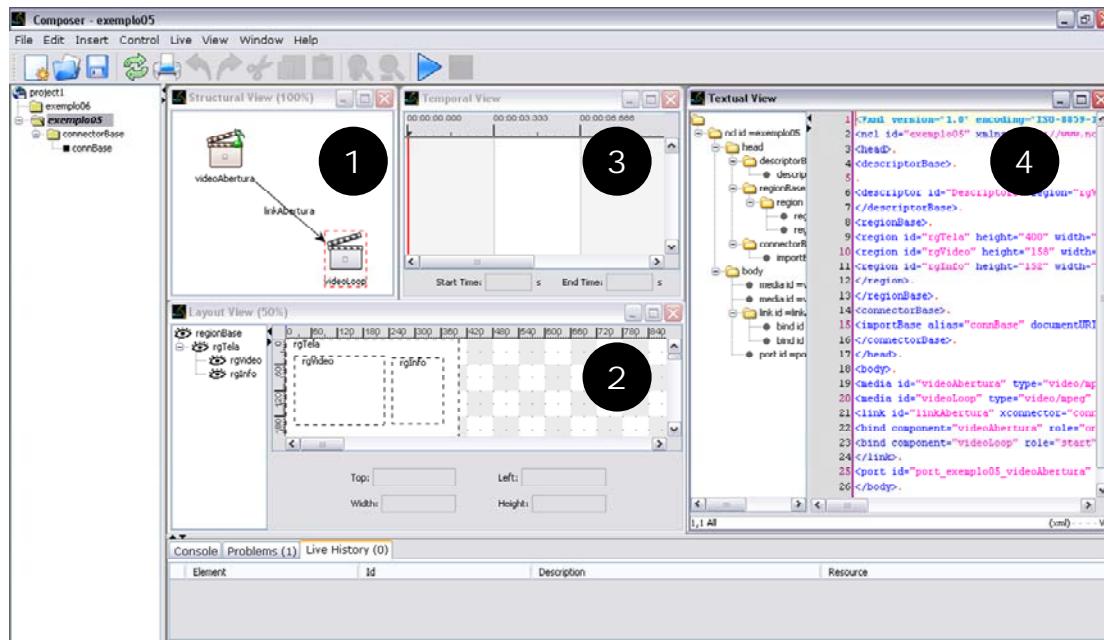


Figura 9. Ferramenta de autoria Composer.

Observa-se na figura que um documento hipermédia pode ser considerado através de diversas visões. A versão atual do Composer permite ao usuário trabalhar com visões Estrutural ①, de Leiaute ②, Temporal ③ e Textual ④.

A **Visão Estrutural** (*Structural View*) apresenta os nós e os elos entre os nós (Figura 10). Nesta visão é possível criar nós de mídia, contextos e elos, bem como definir suas propriedades.

² <http://www.telemidia.puc-rio.br>

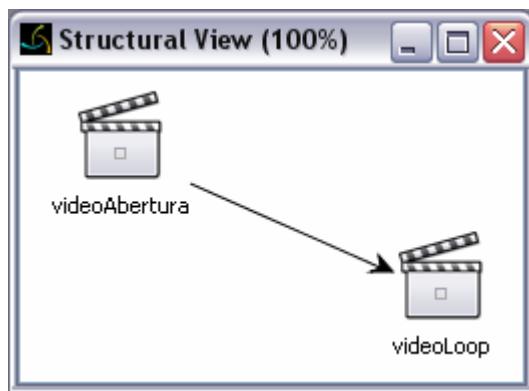


Figura 10. Visão estrutural da ferramenta Composer.

A **Visão Temporal** (*Temporal View*) ilustra o sincronismo temporal entre os nós de mídia, e as oportunidades de interatividade. A Figura 11 apresenta a visão temporal de um documento hipermídia que exibe uma imagem *iconeInteracao* durante um determinado segmento de apresentação da mídia *videoPrinc*.

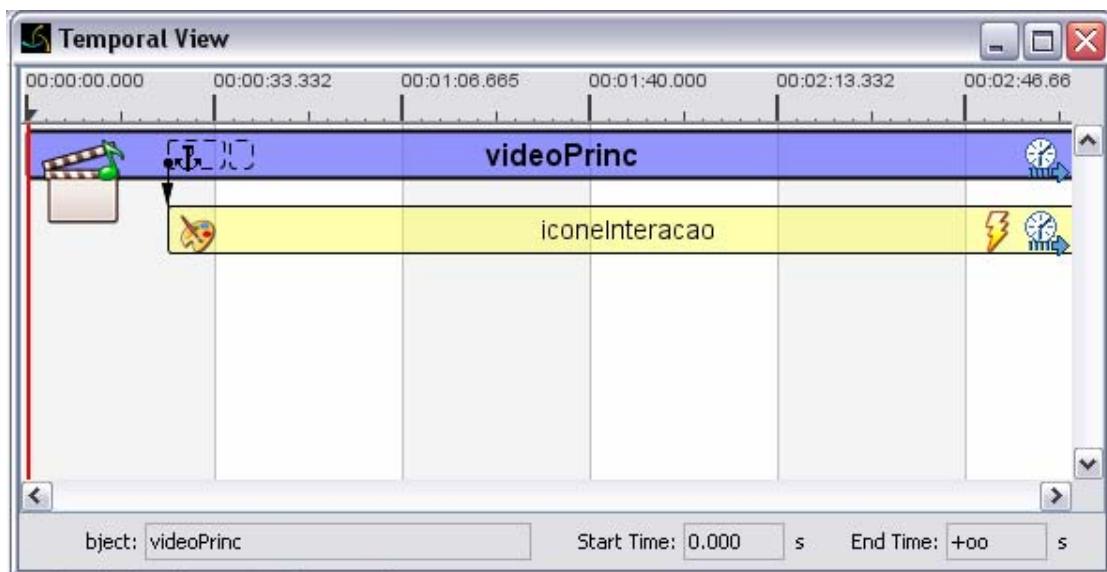


Figura 11. Visão temporal da ferramenta Composer.

A **Visão de Leiaute** (*Layout View*) apresenta as regiões da tela onde as mídias do documento poderão ser apresentadas. A Figura 12 apresenta a visão de leiaute de um documento hipermídia que contém 3 regiões: a tela inteira (*rgTela*), uma área para exibir vídeos (*rgVideo*) e uma área para apresentar informações textuais (*rgTexto*).

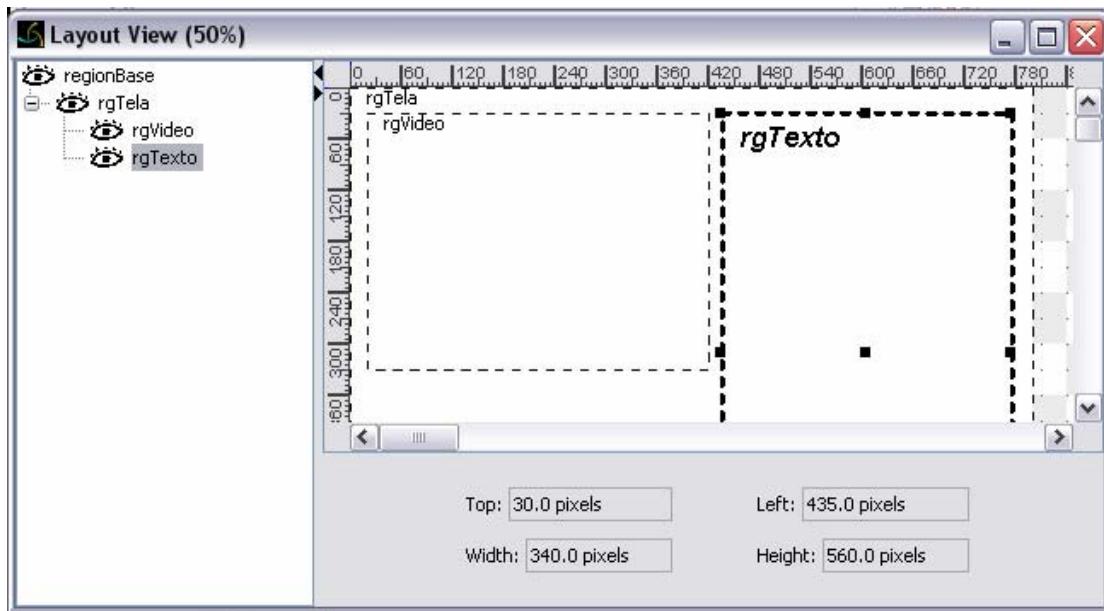


Figura 12. Visão de leiaute da ferramenta Composer.

Já a **Visão Textual** (*Textual View*) apresenta o código NCL em si (Figura 13). Nessa visão, o usuário pode editar diretamente o código NCL como em um editor de texto comum.

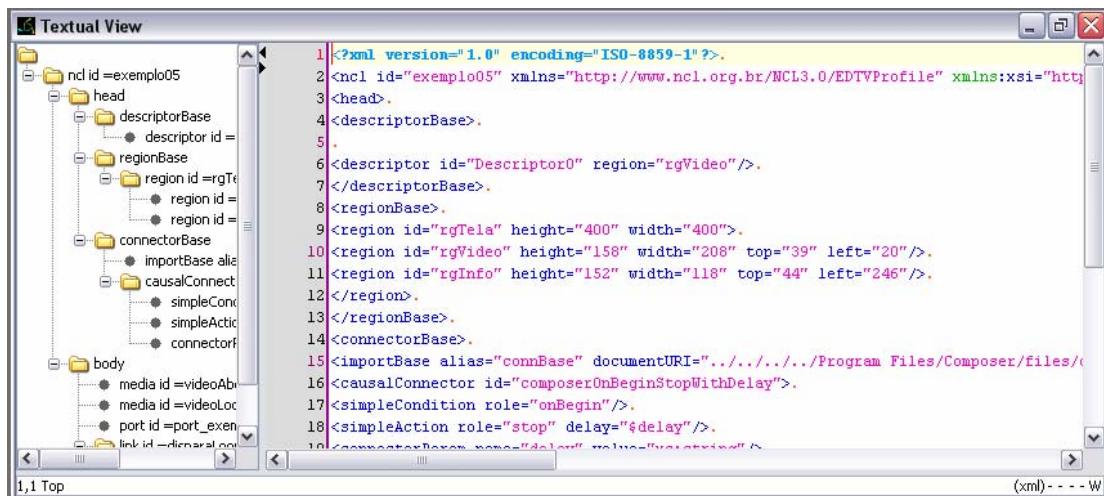


Figura 13. Visão textual da ferramenta Composer.

As visões do Composer são sincronizadas: sempre que são feitas alterações em uma visão, elas são refletidas nas demais visões da ferramenta.

O restante deste documento ensina a construir vários tipos de programa que exploram diferentes recursos da NCL. Cada exemplo é elaborado passo a passo, apresentando-se tanto os passos realizados na ferramenta Composer quanto o código NCL correspondente.

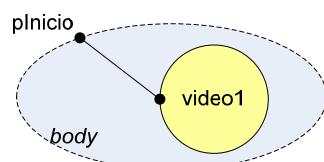
Seu primeiro documento NCL

Exemplo 01 – Reprodução de um objeto de mídia

Esta seção apresenta um documento NCL simples, que apenas reproduz um vídeo no centro da tela de tamanho 1024 x 576 px³. Apesar de não se tratar de um documento hipermídia típico (pois não há nós ligados por elos), o exemplo tem por objetivo familiarizar o leitor com a ferramenta Composer e alguns dos elementos básicos da linguagem NCL.

Visões do documento

Visão estrutural



Visão de leiaute

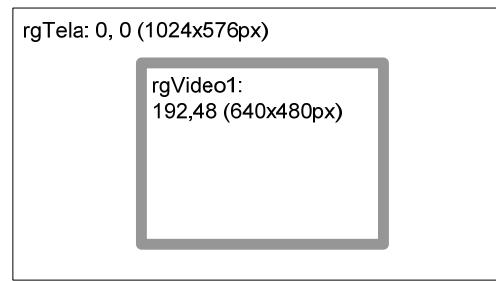


Figura 14. Visões estrutural e de leiaute do exemplo 01, para reprodução de um único vídeo, sem sincronismo ou interação com o usuário.

³ As dimensões de uma TV de alta definição são 1920 x 1080, mas utilizaremos nos exemplos uma dimensão reduzida para os programas poderem ser simulados facilmente em computadores comuns.

Visões temporal e espacial⁴

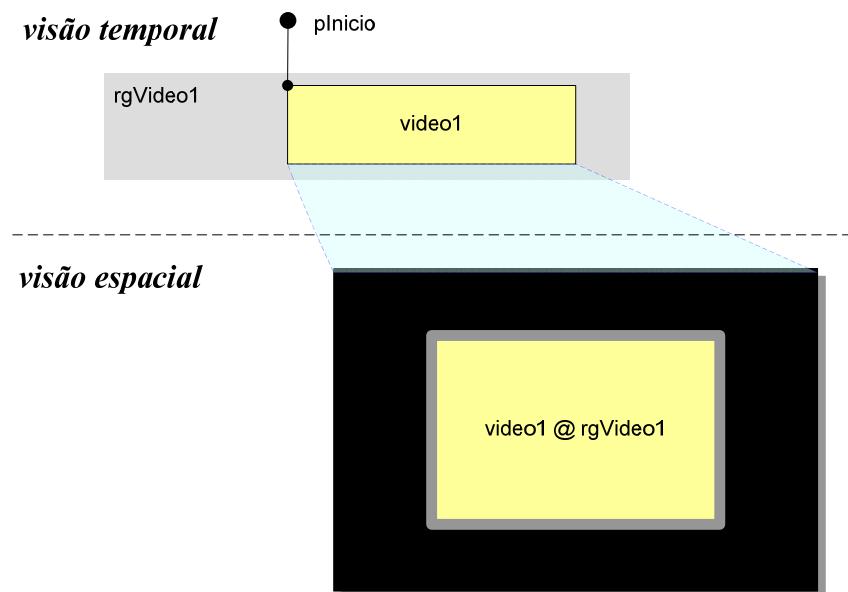


Figura 15. Visões temporal e espacial do exemplo 01, para reprodução de um único vídeo, sem sincronismo ou interação com o usuário.

Passo-a-passo no Composer

Para construir um programa em NCL que reproduz um vídeo, é necessário criar os seguintes elementos:

1. a região da tela que define o dispositivo onde o programa será exibido;
2. a região da tela onde o vídeo será exibido;
3. o descritor que determina a forma como o vídeo será exibido e em que região. Nesse caso, o vídeo associado a este descritor será exibido na região *rgVideo1*, com as propriedades *default* de exibição de vídeo;
4. o nó de mídia *video1*, o vídeo propriamente dito; e
5. a porta que define o ponto de entrada do documento hipermídia. Nesse caso, a porta *pInicio*, que aponta para o primeiro elemento de mídia, *video1*.

Observações:

- Para executar este exemplo, é necessário que haja um vídeo chamado *video1.mpg*.
- A Listagem 1 do Apêndice I apresenta a listagem completa deste exemplo.

⁴ A visão espacial de um documento hipermídia é uma “fotografia” da tela onde o programa audiovisual está sendo apresentado num determinado momento. As visões espaciais aqui representadas visam ilustrar o comportamento dinâmico do programa audiovisual, tal como ele aparece para o telespectador.

Passo 0: Iniciando um novo projeto no Composer

Para começar qualquer projeto no Composer, acione o menu **File > New > Project**, ou pressione **Ctrl+N** (Figura 16).

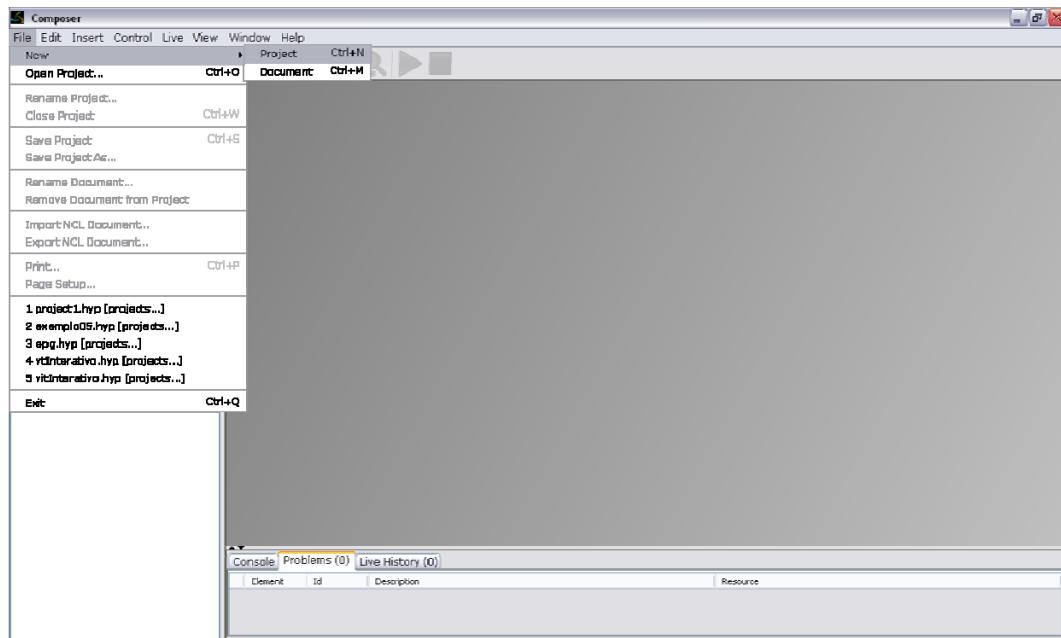


Figura 16. Menu File do Composer.

Surge um quadro de diálogo para especificar o nome e o diretório do projeto (Figura 17). Nesse diretório será armazenado o arquivo do projeto (extensão .hyp). Apesar de os arquivos de mídia do projeto poderem residir em qualquer local, sugerimos que eles sejam mantidos num subdiretório *media* do diretório do projeto.

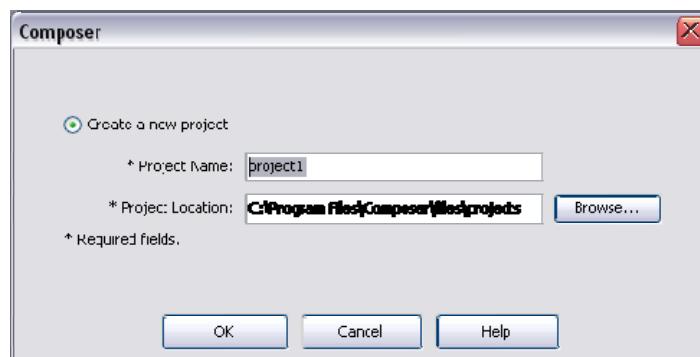


Figura 17. Quadro de diálogo para definição do nome e diretório do projeto.

Caso já exista um projeto com o nome e diretório especificados, o Composer pergunta se pode substituí-lo.

Passos 1 e 2: Definindo regiões de tela

Assim que se abre um novo projeto, recomenda-se especificar as regiões da tela onde as mídias serão apresentadas. Essas regiões são facilmente definidas na visão de leiaute, que pode ser acessada através do menu **View > Layout View** ou pressionando-se Alt+2 (Figura 18):



Figura 18. Menu de acesso à visão de leiaute.

A visão de leiaute apresenta 3 áreas principais (Figura 19): ① área de trabalho; ② árvore de regiões; e ③ painel de propriedades. Essa figura apresenta a visão de leiaute de um novo projeto, ainda sem qualquer região definida.

A **área de trabalho** ① apresenta visualmente as regiões, em dimensões proporcionais, e conforme o nível de zoom corrente. O nível de zoom pode ser alterado clicando-se nos ícones de lupa. A **árvore de regiões** ② apresenta a hierarquia de regiões já definidas e a visibilidade da região na área de trabalho. Uma região pode ser aninhada a outra região, para facilitar a definição de posicionamento e dimensionamento relativos entre regiões. Com relação à visibilidade da região na área de trabalho, um olho aberto indica que a região está visível, ao passo que um olho fechado indica que ela não está visível. Já o **painel de propriedades** ③ apresenta os principais atributos de posição (**left** e **top**) e dimensão (**width** e **height**) da região.

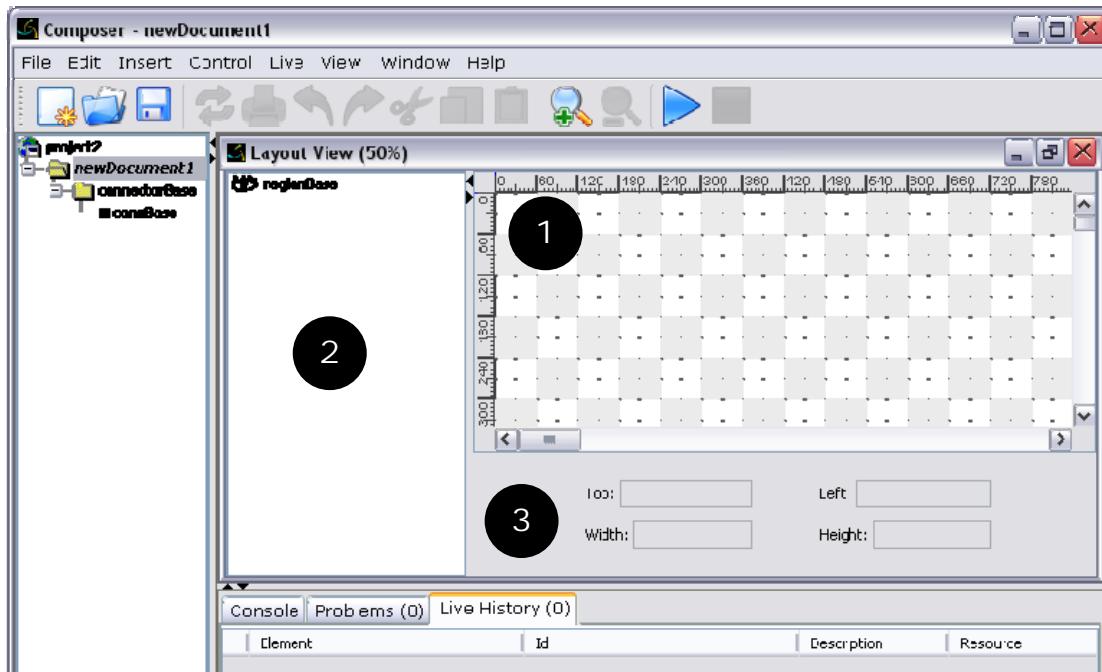


Figura 19. Áreas da visão de layout no Composer.

Para inserir uma região, accese o item de menu **Insert > Region...** (Figura 20a) ou clique com o botão direito do mouse sobre a área de trabalho e selecione **Insert > Region...** no menu pop-up (Figura 20b).

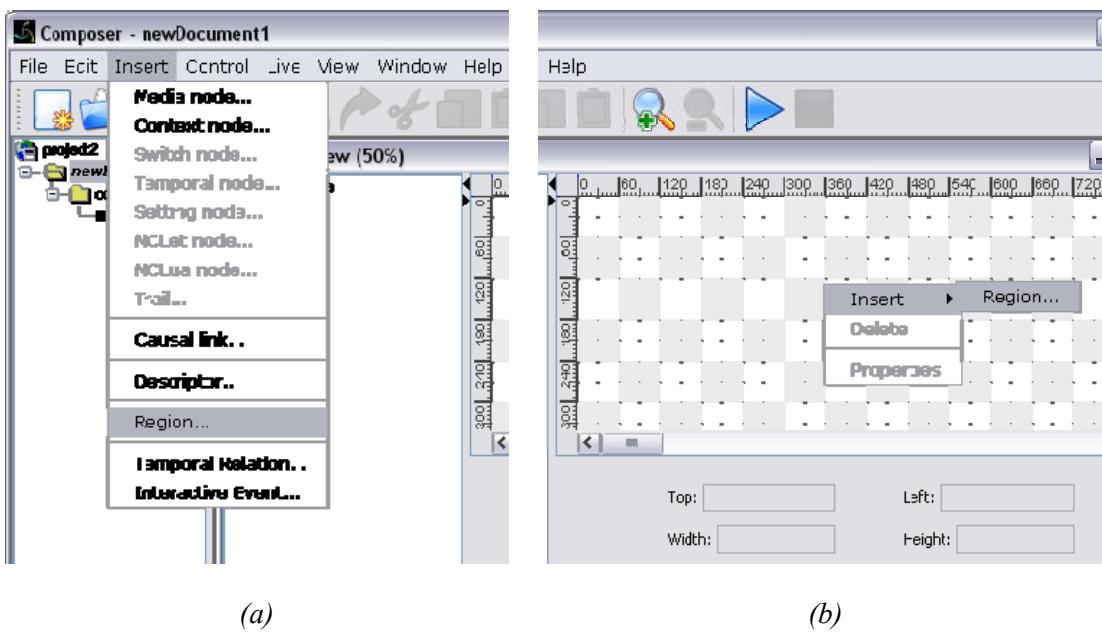


Figura 20. Possibilidades de acesso à criação de uma nova região.

Quando o usuário ativa a operação de **Insert Region**, o Composer apresenta um quadro de diálogo solicitando-lhe que informe os atributos da região. O Composer já define alguns valores *default*, que podem ser modificados pelo usuário (Figura 21a). No caso do exemplo 01, vamos definir que a região do “dispositivo TV”, *rgTV*, mede 1024 x 576 px (Figura 21b).

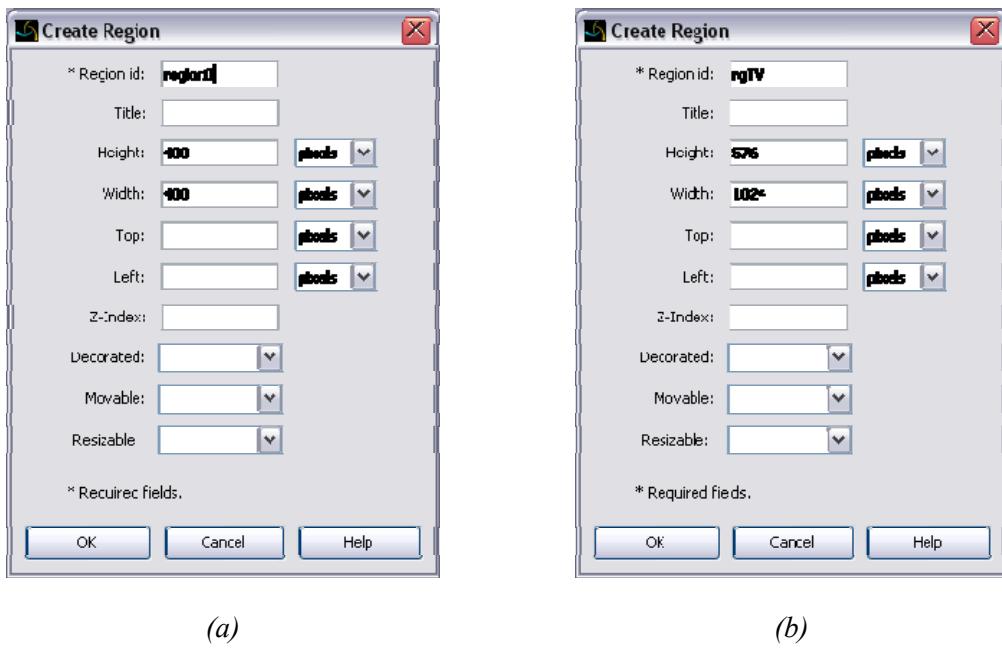


Figura 21. Quadro de diálogo para definição dos atributos de uma região, a) com valores defaults; e b) redefinidos pelo usuário.

Quando o usuário confirma os valores dos atributos, a nova região aparece na área de trabalho e na árvore de regiões (Figura 22), e é selecionada automaticamente. Observa-se ainda que o painel de propriedades exibe a posição e dimensões da região recém-criada, *rgTV*.

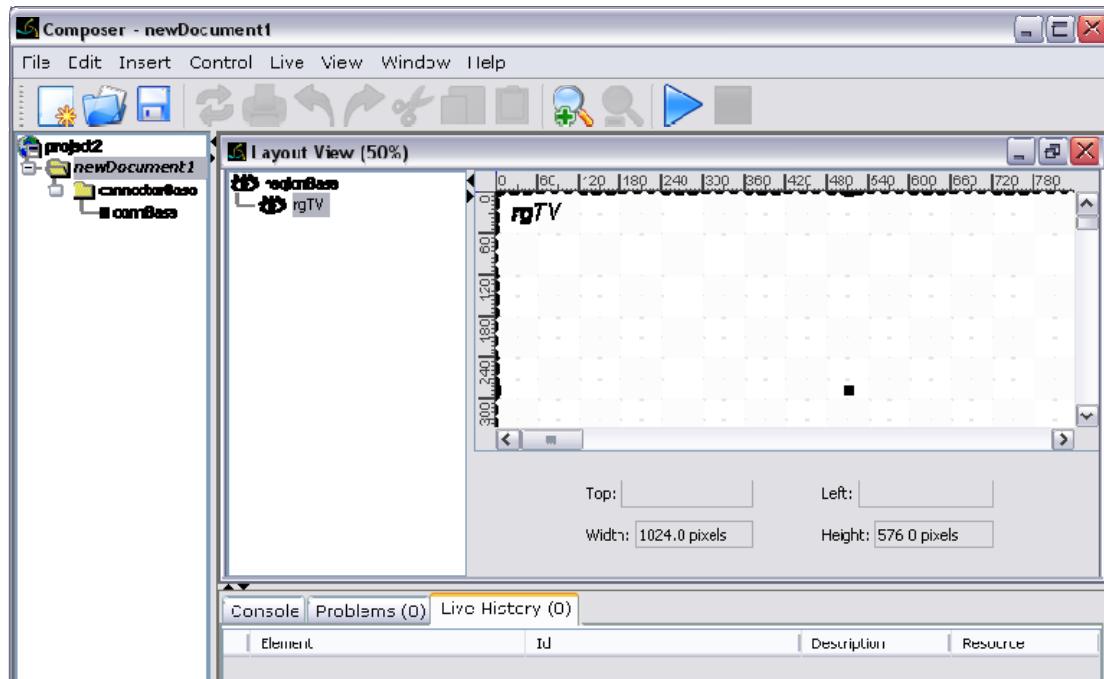


Figura 22. Região recém-criada na visão de leiaute.

Repita a operação de criação de região para criar a região *rgVideo1*, de dimensões 640 x 480 px, onde deve ser exibido o vídeo. Como essa nova região deve ser centralizada na região *rgTV* (posição 192, 48), vamos aninhar a nova região à região *rgTV*. Para isso, basta

selecionar a região *rgTV* (que já deve estar selecionada como resultado da operação anterior de criação de região), e acessar **Insert > Region...**, como da outra vez. Desta vez, preencha não apenas o id e as dimensões da nova região, mas também sua posição, (192px, 48px), como ilustrado na Figura 23.

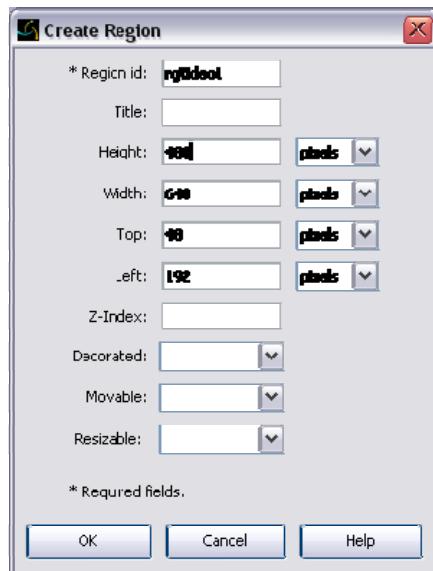


Figura 23. Definição dos atributos da região *rgVideo1*.

Observa-se que a nova região aparece, na área de trabalho e na árvore de regiões, como sendo uma região “filha” da região *rgTV*.

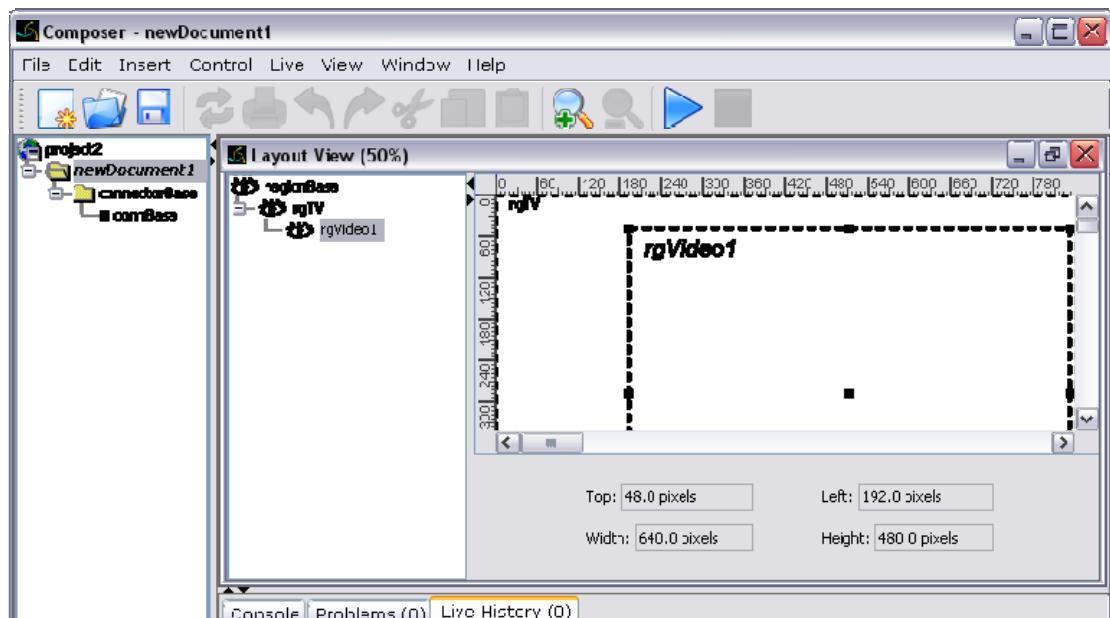


Figura 24. Visão de layout apresentando a região *rgVideo1*, criada dentro da *rgTV*.

É importante observar que todo elemento da NCL (região, descritor, nó de mídia, porta etc.) deve possuir um identificador **único**, representado pelo atributo **id**. Sendo assim, caso o usuário tente criar ou modificar um elemento definindo o **id** de um outro elemento existente

no documento, o Composer emitirá uma mensagem de erro solicitando que seja definido um **id** unívoco.

Para saber mais: Regiões

Uma região (*region*) é definida como uma área no dispositivo de saída onde um nó de mídia pode ser exibido. Para organizar o layout das diversas partes do documento hipermídia, as regiões podem ser aninhadas. 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 serve para inicializar a posição dos nós de mídia num local específico. No exemplo, foram criadas duas regiões, *rgTV*, que define as dimensões do dispositivo, e *rgVideo1*, para que o vídeo fosse apresentado no centro da tela (assumindo uma tela com dimensões 1024x576 pixels e um vídeo com dimensões 640 x 480 pixels):

```
<region id="rgTV" width="1024" height="576">
  <region id="rgVideo1" left="192" top="48" width="640" height="480" />
</region>
```

A NCL define os seguintes atributos de região:

- **id***: identificador único, utilizado nas referências à região (por exemplo, pelos descritores das mídias que serão apresentadas na região)
- **title** (título): título da região. No caso de a região ser exibida com uma moldura, este é o título que aparece como sendo o título da janela correspondente
- **left** (esquerda): coordenada *x* do lado esquerdo da região, com relação à coordenada do lado esquerdo da região pai (ou da tela, no caso de a região não estar aninhada a nenhuma outra)
- **top** (topo): coordenada *y* do lado superior da região, com relação à coordenada do lado superior da região pai (ou da tela, no caso de a região não estar aninhada a nenhuma outra)
- **right** (direita): coordenada *x* do lado direito da região, com relação à coordenada do lado direito da região pai (ou da tela, no caso de a região não estar aninhada a nenhuma outra)
- **bottom** (base): coordenada *y* do lado inferior da região, com relação à coordenada do lado inferior da região pai (ou da tela, no caso de a região não estar aninhada a nenhuma outra)

* Os atributos marcados com asterisco são obrigatórios.

- **width** (largura) e **height** (altura): dimensões horizontal e vertical da região. Cabe observar que o autor pode escolher especificar as dimensões de uma região conforme sua conveniência. Por exemplo, em certos casos pode ser melhor definir os atributos **right**, **bottom**, **width** e **height**. Em outros casos, pode ser mais apropriado especificar os atributos **top**, **left**, **width** e **height**.
- **zIndex**: posição da região no eixo “z”, utilizado geralmente para indicar, no caso de regiões sobrepostas, quais regiões aparecem sobre quais outras. Camadas com **zIndex** maior são apresentadas no topo de (sobrepondo) camadas com **zIndex** menor.

A Figura 25 ilustra os atributos **top**, **left**, **right**, **bottom**, **width**, **height** e **zIndex**:

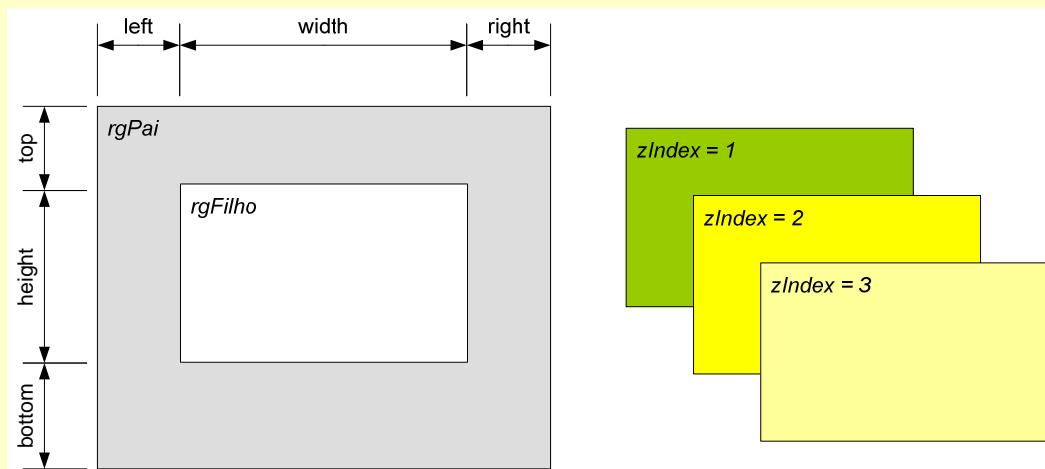


Figura 25. Atributos de posicionamento e dimensão de uma região.

É importante observar que, caso todos os atributos de posicionamento e dimensão sejam especificados, os atributos **left** e **width** têm precedência sobre o atributo **right**, assim como os atributos **top** e **height** têm precedência sobre o atributo **bottom**.

Além disso, quando duas ou mais regiões possuírem o mesmo valor de **zIndex**, e em cada região for apresentada uma mídia, existem duas possibilidades: caso uma mídia seja apresentada antes da outra, a mídia que for iniciada depois vai se sobrepor à que foi iniciada antes (ordem temporal). Caso as duas mídias sejam iniciadas ao mesmo tempo, a ordem é escolhida aleatoriamente pelo formatador.

Já a base de regiões possui os seguintes atributos:

- **id**: identificador único da base de regiões
- **device**: dispositivo ao qual a região está associada. Pode ser da forma systemScreen(i), systemAudio(i), conforme os dispositivos disponíveis.

Ao final desse passo, pode-se observar, na visão textual, o código NCL correspondente às regiões criadas:

```
<regionBase>
```

```
<region id="rgTV" height="576" width="1024">
    <region id="rgVideo1" height="480" width="640" top="48" left="192"/>
</region>
</regionBase>
```

Observa-se, no código NCL, o aninhamento da região *rgVideo1* na região *rgTV*.

Passo 3: Definindo o descriptor que determina como o vídeo será exibido

Tendo especificado as regiões no Composer, pode-se agora definir o descriptor que determinará como o vídeo será exibido. No caso do exemplo 01, o descriptor será utilizado apenas para associar uma mídia a uma região.

Acesse o item de menu **Insert > Descriptor...** para iniciar a criação do descriptor (Figura 26a). O Composer apresentará um quadro de diálogo solicitando informar os parâmetros do descriptor. Defina o **id** do descriptor como sendo *dVideo1* (Figura 26b) e confirme.

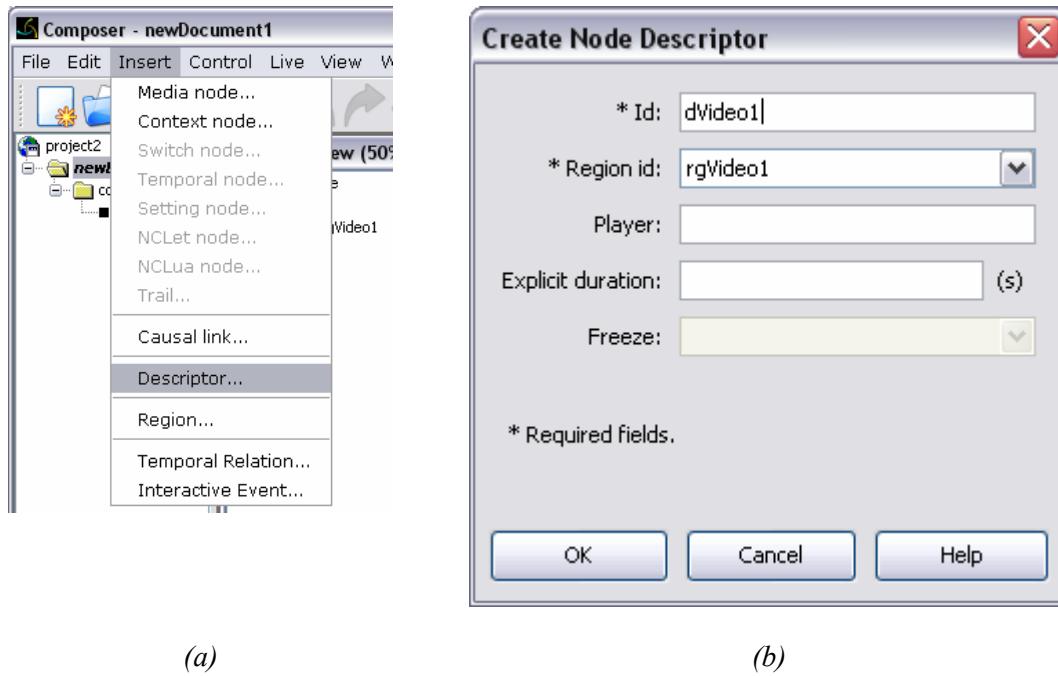


Figura 26. Criação de um descriptor para a região *rgVideo1*.

Para saber mais: Descritores

Um descritor (*descriptor*) define como um nó de mídia será apresentado, incluindo a associação com uma região onde será exibido. Em NCL, todos os descritores devem ser definidos no cabeçalho do programa (**<head>**), na seção de base de descritores (**<descriptorBase>**). Todo nó de mídia que será apresentado deve ter um descritor associado.

No exemplo, foi criado um descritor *dVideo1* se referindo à região *rgVideo1*. A NCL define os seguintes atributos de descritor:

- **id***: identificador único, utilizado nas referências ao descritor (por exemplo, nos nós de mídia/conteúdo apresentados pelo descritor)
- **player**: identifica a ferramenta de apresentação a ser utilizada para exibir o objeto de mídia associado ao descritor. Esse atributo é opcional. Quando omitido, o formatador procura pela ferramenta *default* em função do tipo do objeto de mídia.
- **explicitDur**: define a duração ideal do objeto de mídia associado ao descritor. O valor deste atributo deve ser expresso em segundos, no formato 9.9s (valor numérico seguido da letra “s”). Quando **explicitDur** não for especificado, o objeto que estiver associado ao descritor será exibido com sua duração *default*. Mídias como textos e imagens estáticas possuem duração *default* infinita. Na implementação de referência do formatador, esse atributo não é considerado em mídias contínuas. Em um formatador com suporte ao ajuste elástico de tempo, o valor desse atributo é usado como referência para modificar o tempo de duração do nó (permitindo, por exemplo, que alguns quadros de um vídeo de 30s deixem de ser exibidos para que ele dure 29s).
- **region**: identificador da região associada ao descritor. Ao utilizar o descritor, o objeto de mídia será exibido na região correspondente. Esse atributo só precisa ser especificado para objetos que possuam um conteúdo visual a ser exibido.
- **freeze**: identifica o que acontece ao final da apresentação do objeto de mídia associado ao descritor. Em um vídeo, o valor “true” indica que o último quadro deve ser congelado.
- **focusIndex**: define um índice de navegação para o objeto de mídia associado ao descritor. Caso não seja definido um índice, o objeto de mídia não poderá receber o foco da navegação. O foco inicial estará no objeto com **focusIndex** menor. Vale observar que o valor de **focusIndex** não é necessariamente um número e, nesse caso, “menor” significa lexicograficamente menor.
- **focusBorderColor**: define a cor do retângulo que deve aparecer sobreescrito à região deste descritor quando o objeto a ele associado receber o foco. Pode ser um dos seguintes valores: *white, black, silver, gray, red, maroon, fuchsia, purple, lime, green, yellow, olive, blue, navy, aqua, ou teal*.
- **focusBorderWidth**: define a espessura, em *pixels*, do retângulo que deve aparecer sobreescrito à região deste descritor quando o elemento a ele associado receber o foco. Caso seja igual a 0, nenhuma borda será apresentada. Um valor positivo indica que a borda estará fora do conteúdo do objeto, ao passo que um valor

negativo indica que a borda será desenhada sobre o conteúdo do objeto.

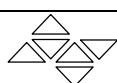
- **focusBorderTransparency:** define a porcentagem de transparência da borda. Deve ser um valor real entre 0 e 1, onde 0 significa totalmente opaco e 1 significa totalmente transparente.
- **focusSrc:** define uma mídia alternativa que deve ser exibida quando o elemento associado a este descritor estiver com o foco
- **focusSelSrc:** define uma mídia alternativa que deve ser exibida quando for pressionado o botão OK ou Enter enquanto o elemento associado a este descritor estiver com o foco
- **selBorderColor:** define uma cor de borda que deve ser exibida quando for pressionado o botão OK ou Enter enquanto o elemento associado a este descritor estiver com o foco
- **moveLeft:** identifica o índice de navegação do elemento E que deve receber o foco caso seja pressionada a seta para a esquerda do controle remoto enquanto o elemento associado a este descritor estiver com o foco (definido pelo atributo **focusIndex** do elemento E)
- **moveRight:** identifica o índice de navegação do elemento E que deve receber o foco caso seja pressionada a seta para a direita do controle remoto enquanto o elemento associado a este descritor estiver com o foco (definido pelo atributo **focusIndex** do elemento E)
- **moveUp:** identifica o índice de navegação do elemento E que deve receber o foco caso seja pressionada a seta para cima do controle remoto enquanto o elemento associado a este descritor estiver com o foco (definido pelo atributo **focusIndex** do elemento E)
- **moveDown:** identifica o índice de navegação do elemento E que deve receber o foco caso seja pressionada a seta para baixo do controle remoto enquanto o elemento associado a este descritor estiver com o foco (definido pelo atributo **focusIndex** do elemento E)
- **transIn:** define a transição que será executada ao iniciar a apresentação do objeto de mídia
- **transOut:** define a transição que será executada ao terminar a apresentação do objeto de mídia

A NCL define ainda o seguinte elemento opcional contido num elemento de descritor:

- **descriptorParam:** define um parâmetro do descritor como um par <propriedade, valor>. As propriedades e seus respectivos valores dependem do programa de exibição da mídia associada ao descritor.

Cada descritor pode conter diversos elementos **descriptorParam**, definidos no formato:

```
<descriptorParam name="nome_do_parametro" value="valor_do_parametro" />
```



para indicar que a mídia correspondente deve ser reproduzida com o volume a 90% do máximo:

```
<descriptor id="dVideo1" region="rgVideo1">
  <descriptorParam name="soundLevel" value="0.9" />
</descriptor>
```

O uso de parâmetros de descritor promove um alto grau de flexibilidade. Cabe a cada programa de exibição do objeto de mídia (**player**) interpretar essas propriedades da forma adequada. Atualmente, não se pode definir parâmetros de descritor no Composer. Em NCL, estão reservados os seguintes parâmetros:

Tabela 2. Parâmetros que podem ser utilizados em descriptorParam, conforme a mídia.

tipo de mídia	parâmetro	Descrição
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)
objetos visuais	location	posição do objeto de mídia. Trata-se de dois números separados por vírgula, na ordem <left, top> num dos seguintes formatos: a) números reais entre 0 e 100, seguidos do sinal de porcentagem; ou b) números inteiros não negativos que especifiquem um valor em <i>pixels</i> .
objetos visuais	size	dimensões do objeto de mídia. Trata-se de dois números separados por vírgula, na ordem <width, height> , num dos seguintes formatos: a) números reais entre 0 e 100, seguidos do sinal de porcentagem; ou b) números inteiros não negativos que especifiquem um valor em <i>pixels</i> .
objetos visuais	bounds	posição e dimensões do objeto de mídia. Trata-se de quatro números separados por vírgula, na ordem <left, top, width, height> , num dos seguintes formatos: a) números reais entre 0 e 100, seguidos do sinal de porcentagem; ou b) números inteiros não negativos que especifiquem um valor em <i>pixels</i> .
objetos visuais	background	nomes de cores reservados: <i>white, black, silver, gray, red, maroon, fuchsia, purple, lime, green, yellow, olive, blue, navy, aqua</i> , ou <i>teal</i> . Também pode ser <i>transparent</i> , para o caso de

imagens com transparência, como alguns GIFs.		
objetos visuais	visible	<i>true</i> ou <i>false</i>
objetos visuais	transparency	número real entre 0 e 1 indicando transparência: 0 significa totalmente opaco e 1 significa totalmente transparente
objetos visuais	fit	um dos seguintes valores: <i>fill</i> , <i>hidden</i> , <i>meet</i> , <i>meetBest</i> ou <i>slice</i> , onde: <ul style="list-style-type: none"> ▪ <i>fill</i> = redimensiona o conteúdo do objeto de mídia para que toque todas as bordas da região; ▪ <i>hidden</i>: se a altura intrínseca ao conteúdo da mídia for menor que o atributo height, o objeto precisa ser renderizado a partir do topo e ter sua altura restante preenchida com a cor de background; caso seja maior, o restante deve ser cortado. Idem para a largura e esquerda. ▪ <i>meet</i> = redimensiona o conteúdo do objeto de mídia mantendo suas proporções até atingir uma das bordas da região. Caso haja um espaço vazio à direita ou na parte de baixo, deve ser preenchido com a cor de background. ▪ <i>meetBest</i> = semelhante ao <i>meet</i>, mas o objeto de mídia não é ampliado em mais do que o dobro das dimensões originais ▪ <i>slice</i> = redimensiona o conteúdo do objeto de mídia mantendo suas proporções até que toda a região seja preenchida. Parte do conteúdo pode ser cortado à direita ou na parte de baixo do conteúdo.
objetos visuais	scroll	um dos seguintes valores: <i>none</i> , <i>horizontal</i> , <i>vertical</i> , <i>both</i> ou <i>automatic</i>
objetos visuais	style	localização de um arquivo de folha de estilo

Os descritores não aparecem nas visões estrutural, de leiaute ou temporal no Composer. Para verificar se um descritor foi criado corretamente, deve-se consultar a visão textual. No caso desse exemplo, o código NCL correspondente ao descritor criado é o seguinte:

```
<descriptorBase>
  <descriptor id="dVideo1" region="rgVideo1"/>
</descriptorBase>
```

Passo 4: Definindo a mídia propriamente dita

Tendo definido as regiões e os descritores, o próximo passo é criar os objetos de mídia que compõem o documento. Para organizar os objetos de mídia no documento, recomenda-se utilizar o Composer na visão estrutural (acessada pelo item de menu **View > Structural View**, conforme ilustrado na Figura 27).

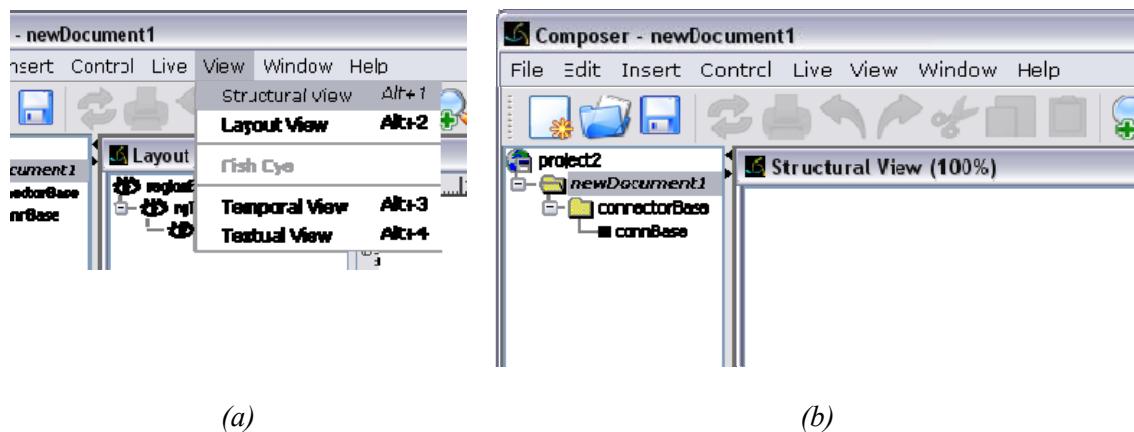
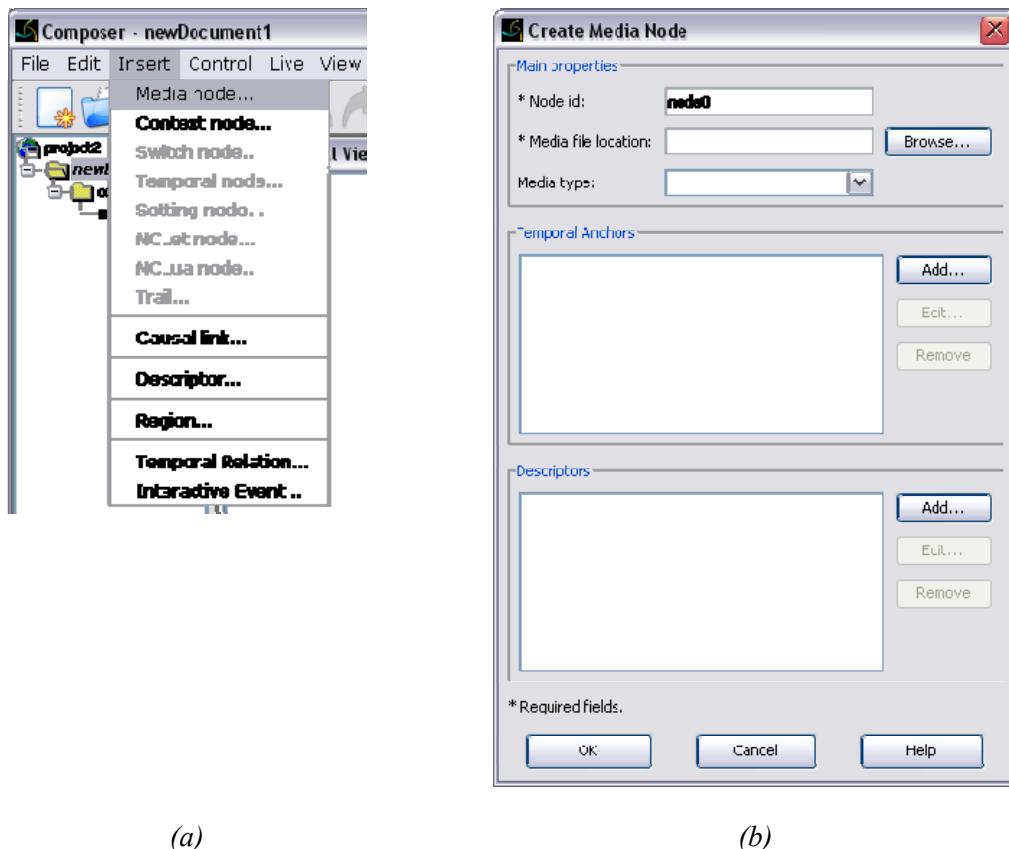


Figura 27. Visão estrutural: a) acesso e b) visão estrutural vazia.

Você pode observar que a visão estrutural ainda está vazia. Para criar um nó de mídia, acesse o item de menu **Insert > Media node...** (Figura 28a). O Composer apresenta um quadro de diálogo com os atributos de nó de mídia (Figura 28b).



(a)

(b)

Figura 28. Criando um nó de mídia: a) acesso via menu; b) quadro de diálogo dos atributos de um nó de mídia.

Defina um nó de mídia com nome *video1*, localize um arquivo do tipo MPEG no seu computador e selecione o tipo *video/mpeg* (Figura 29).

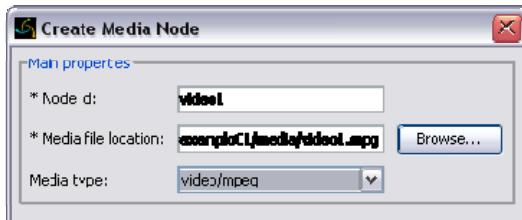


Figura 29. Atributos do nó de vídeo do exemplo 01.

O próximo passo, ainda nesse quadro de diálogo, é associar um descritor a esse nó de mídia. Para isso, clique em **Add** no grupo **Descriptors**. O Composer apresenta o seguinte quadro de diálogo (Figura 30):



Figura 30. Quadro de diálogo para associação de um descritor ao nó de mídia.

Como o descritor *dVideo1* já foi criado, marque **Select an existing descriptor** e confirme. Em seguida, confirme a criação do nó, no quadro **Create Media Node**.

Observe que, na visão estrutural, apareceu um ícone representando o objeto de mídia recém criado (Figura 31).

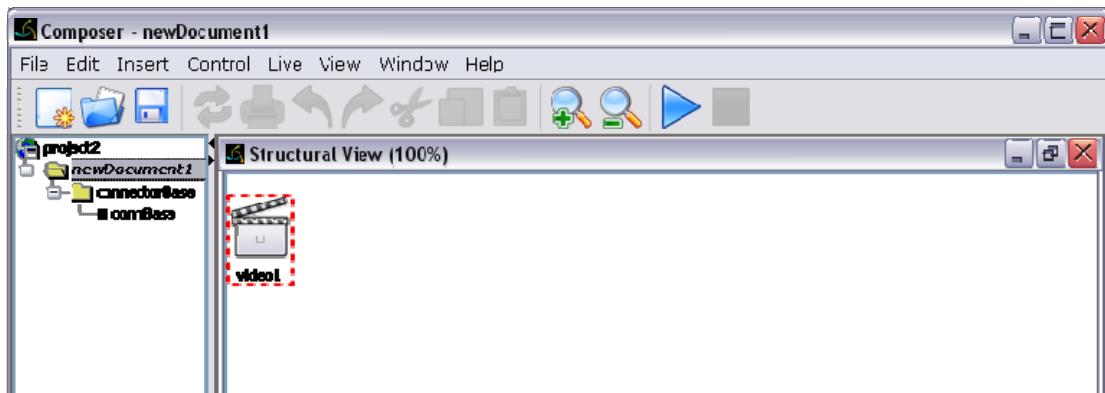


Figura 31. Visão estrutural exibindo o nó de mídia recém criado, *video1*.

Para saber mais: Nó de mídia (ou nó de conteúdo)

Um nó de mídia ou de conteúdo (*media node* ou *content node*) define o objeto de mídia propriamente dito: vídeo, áudio, texto etc. Cada definição de nó de mídia deve apresentar, além do arquivo de origem da mídia, o descritor que regulará a apresentação daquele objeto de mídia. No exemplo anterior, foi definido apenas um nó de mídia, do tipo vídeo, chamado *video1*:

Um nó de mídia possui os seguintes atributos:

- **id***: identificador único do nó de mídia, utilizado nas referências ao nó (por exemplo, nas portas dos nós de contexto que levam à mídia)
- **type**: tipo de mídia, especificado como MIME, conforme a seguinte tabela:

valor de type	extensão de arquivo do atributo src
text/html	.htm, .html
text/css	.css
text/xml	.xml
image/bmp	.bmp
image/png	.png
image/gif	.gif
image/jpeg	.jpg
audio/basic	.wav
audio/mp3	.mp3
audio/mp2	.mp2
audio/mpeg4	.mp4, .mpg4
video/mpeg	.mpeg, .mpg
application/x-ginga-NCLua	.lua
application/x-ginga-NCLet	.xlt, .xlet, .class
application/x-ginga-settings	Não tem arquivo associado (não se define o atributo src). Trata-se de um nó de atributos globais para ser utilizado em regras e <i>switches</i> .
application/x-ginga-time	Não tem arquivo associado (não se define o atributo src)

- **src**: fonte do objeto de mídia. Trata-se do caminho para o arquivo de mídia. Esse caminho pode ser relativo, a partir do diretório onde se encontra o arquivo NCL, ou absoluto, através de uma URI. As URI válidas são as seguintes:

esquema	formato	uso
file:	//caminho_arquivo/#id_fragmento	arquivos locais
http:	//id_servidor/caminho_arquivo/#id_fragmento	arquivos remotos baixados do canal de retorno utilizando o protocolo http
rstp:	//id_servidor/caminho_arquivo/#id_fragmento	streams baixados do canal de retorno utilizando o protocolo rstp
rtp:	//id_servidor/caminho_arquivo/#id_fragmento	streams baixados do canal de retorno utilizando o protocolo rtp
isdtv-ts:	//id_programa	streams recebidos da transport stream

- **descriptor**: identificador do descritor que controla a apresentação do objeto de mídia
- **refer**: referência a um outro nó de mídia previamente definido, como forma de re-

uso de nó (utiliza os atributos do nó de mídia referenciado, exceto o **id**)

- **newInstance**: estabelece se um nó que se refira a outro gera uma nova instância do objeto no formatador ou se reutiliza a instância previamente criada

Na visão textual, tem-se o seguinte código NCL correspondente ao nó *video1* criado:

```
<media id="video1" type="video/mpeg" src="media/video1.mpg" descriptor="dVideo1"/>
```

Além disso, ainda na visão textual, observa-se que esse nó de mídia está dentro do elemento body:

```
<body>
  <media id="video1" type="video/mpeg" src="media/video1.mpg" descriptor="dVideo1"/>
  ...
</body>
```

Para saber mais: Contextos

O elemento **body** é um caso particular de contexto, representando o documento como um todo. Contextos ou nós de composição são utilizados para estruturar um documento hipermídia. Os contextos podem ser aninhados, para refletir a estrutura do documento e ajudar o autor a organizar os segmentos do programa audiovisual interativo.

Um contexto é definido da seguinte forma:

```
<context id="ctxNome">
  ...
</context>
```

Os atributos de um contexto são:

- **id**: identificador único do contexto
- **refer**: referência a um outro contexto previamente definido (utiliza os atributos do contexto referenciado, exceto o **id**)

O contexto **body** de um documento NCL herda o **id** do próprio documento.

Passo 5: Definindo a porta do contexto *body* que determina “onde” o programa inicia (apresentando qual mídia)

Para concluir a criação desse primeiro exemplo, é necessário definir pelo menos uma porta do contexto *body* que mapeia para um nó de mídia. O Composer oferece um atalho para fazer isso: clique com o botão direito do mouse sobre o nó de mídia *video1* e selecione, no menu pop-up, o item **Set as Starting Node** (Figura 32a). Observe que o nó, na visão estrutural, é decorado com um ícone que representa o nó com porta no contexto *body* (Figura 32b).

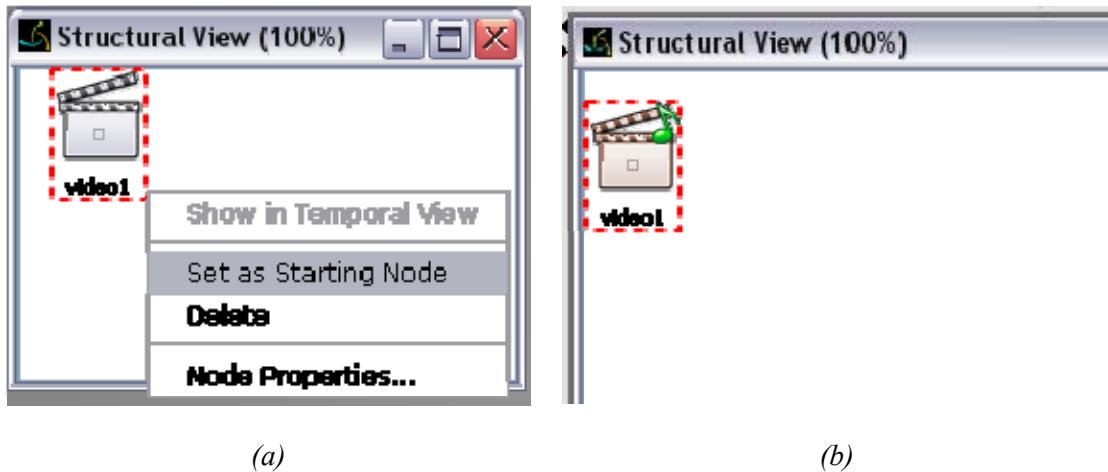


Figura 32. (a) Definindo uma porta para o nó de mídia e (b) indicação visual do nó com porta no contexto body.

Para saber mais: Portas

Uma porta (*port*) é um ponto de interface (*interface point*) de um contexto, que oferece acesso externo ao conteúdo (nós internos) de um contexto. Em outras palavras, para um elo apontar para um nó interno ao contexto, este contexto deve possuir uma porta que leve ao nó interno desejado (Figura 33).

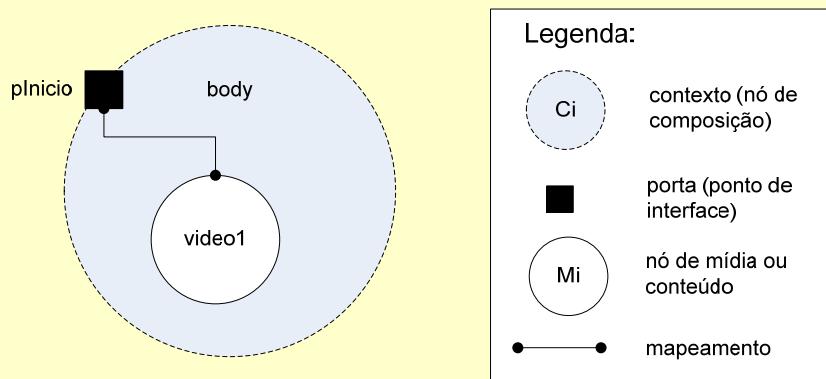


Figura 33. Porta pInicio como ponto de entrada a um nó interno de um contexto.

Em todo documento, deve haver pelo menos uma porta de entrada (**port** na seção **body**) indicando qual é o nó de mídia ou contexto inicial.

Observe que a porta *pInicio* do *body* mapeia para o vídeo *video1*. Isto significa que, ao iniciar o documento hipermídia, o formatador seguirá a porta *pInicio*, que leva ao nó de conteúdo *video1*, que será então apresentado.

Uma porta possui os seguintes atributos:

- **id***: identificador único da porta, utilizado nas referências à porta (por exemplo, por elos)

- **component***: nó de mídia ou contexto ao qual a porta mapeia.
Caso **component** seja um contexto, deve-se definir ainda o atributo **interface**, fazendo o mapeamento para uma porta ou âncora daquele contexto.
Caso **component** seja um nó de mídia, pode-se definir o atributo **interface** apenas como sendo uma âncora do nó de mídia. Se o atributo **interface** for omitido, todo o nó será considerado como mapeado àquela porta.
- **interface**: nome do ponto de interface (porta) de destino no contexto ou nome da âncora de destino no nó de mídia ou contexto

Voltando à visão textual, observa-se que o código NCL correspondente à porta definida no exemplo é o seguinte:

```
<port id="port_newDocument1_video1" component="video1"/>
```

Passo 6: Testando o documento

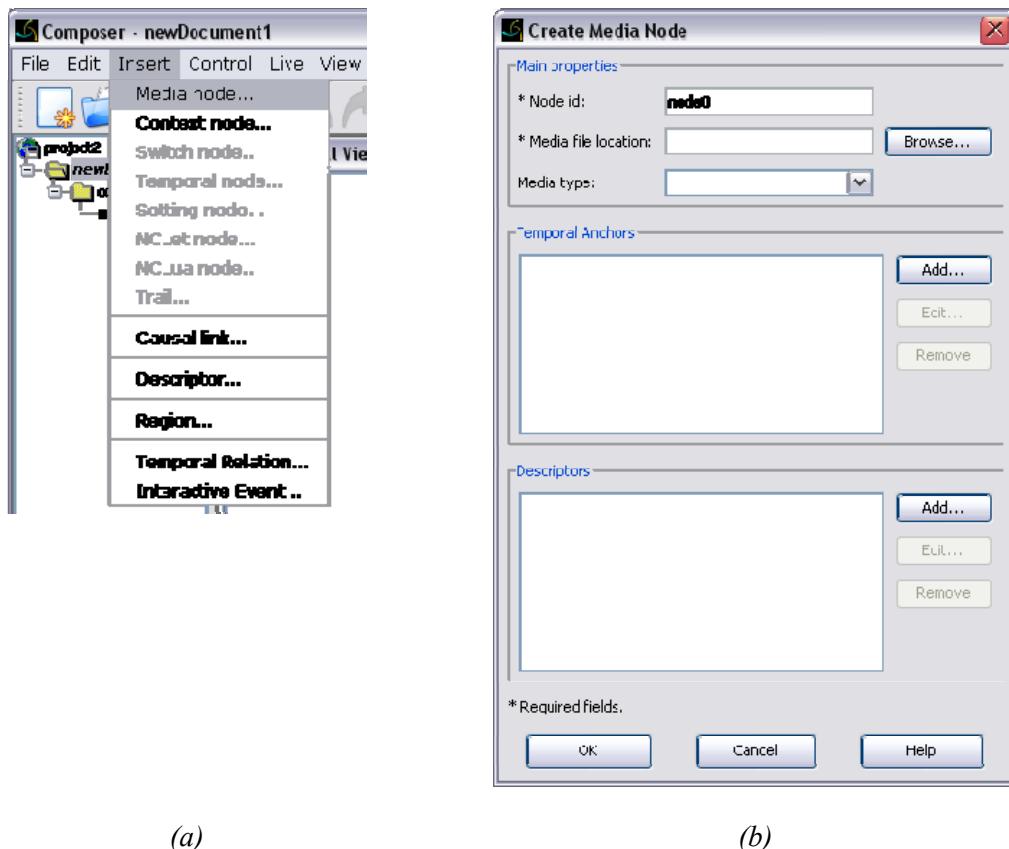
Para testar o documento, utilize o item de menu **Control > Play**, ou clique sobre o botão na barra de ferramentas.

Passo-a-passo alternativo para a criação de nós, regiões e descritores

No passo-a-passo anterior, optamos por criar primeiro a região, depois o descritor, e depois o nó. No entanto, uma estratégia comum de criação de documentos hipermídia consiste em definir primeiramente um nó, seguido da região onde será apresentado e somente depois o descritor que faz essa associação. O Composer facilita essa estratégia, encaminhando o autor, após a definição de um nó, para a definição ou seleção da região, e criando automaticamente um descritor para fazer a associação entre o nó e a região desejada. As figuras a seguir ilustram esse processo.

Passo 1: Criando um nó de mídia

Para criar um nó de mídia, acesse o item de menu **Insert > Media node...** (Figura 42a). O Composer apresenta um quadro de diálogo com os atributos de nó de mídia (Figura 42b).



(a)

(b)

Figura 34. Criando um nó de mídia: a) acesso via menu; b) quadro de diálogo dos atributos de um nó de mídia.

Defina um nó de mídia com nome *video1*, localize um arquivo do tipo MPEG no seu computador e selecione o tipo *video/mpeg* (Figura 35).

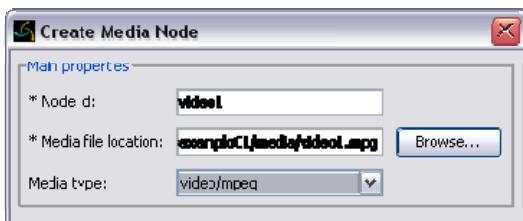


Figura 35. Atributos do nó de vídeo do exemplo 01.

Clique em OK no quadro **Create Media Node**. O Composer exibe um quadro de mensagem perguntando se você deseja associar uma região a esse nó (Figura 36):

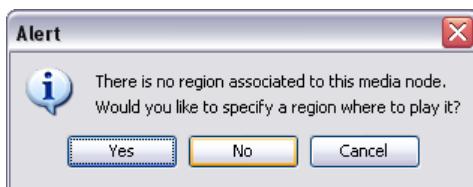


Figura 36. Quadro de diálogo para associar uma região ao nó recém-criado.

Passo 2: Criando ou selecionando uma região

Ao responder que sim, você inicia o segundo passo, de criação ou seleção de região a ser associada ao nó que acaba de criar.



Figura 37. Quadro de diálogo para associação de uma região a um nó, com as opções de criar uma nova região ou selecionar uma região existente.

Caso a região não exista, clique em OK para criá-la. O Composer apresenta então o quadro de diálogo para a definição da nova região, cujo identificador default é composto a partir do identificador do nó recém-criado (Figura 38).

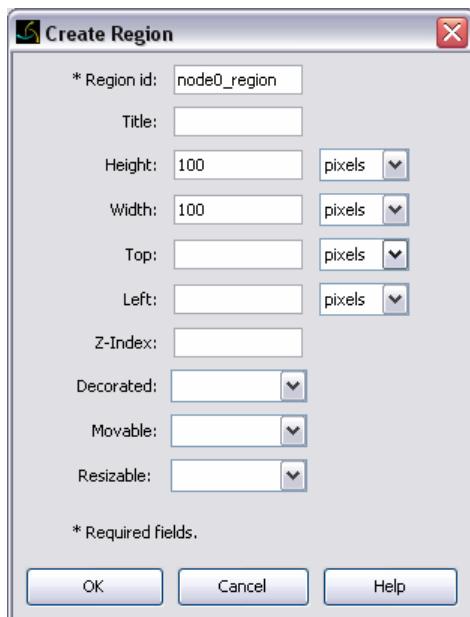


Figura 38. Quadro de diálogo para definição de região.

Após editar os campos para definir a região com os atributos desejados, clique em OK. O Composer apresenta o novo nó na visão estrutural (Figura 39).

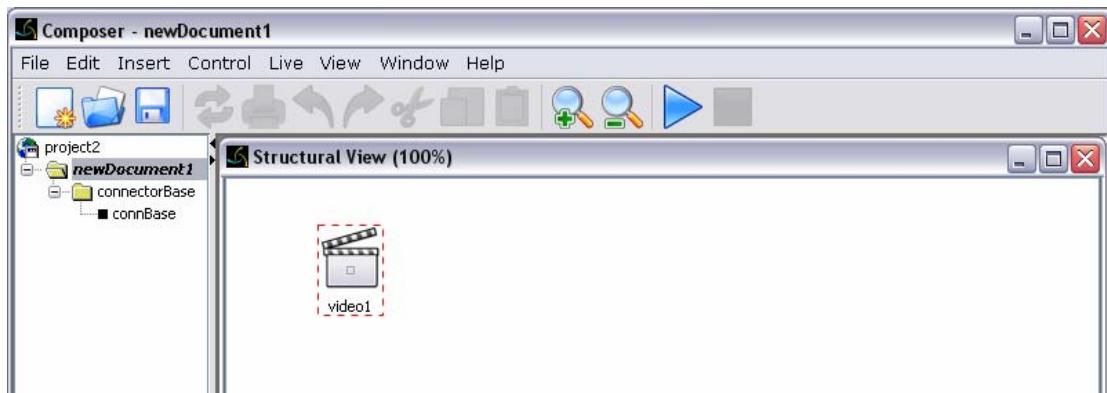


Figura 39. Visão estrutural apresentando o nó recém-criado.

Passo 3: Criando ou modificando um descritor

O Composer criou automaticamente um descritor que faz a associação entre o nó criado e a região criada ou selecionada. Para editar esse descritor, você pode seguir dois caminhos:

- utilizar o menu **Edit > Descriptors...** e selecionar o descritor desejado (Figura 40); ou

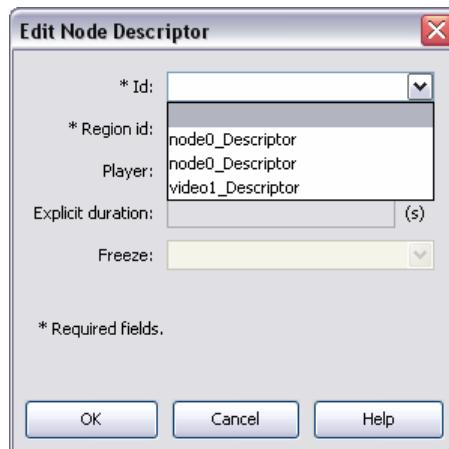


Figura 40. Quadro para edição de descritor a partir do menu **Edit > Descriptors....**

- clicar com o botão direito sobre o nó, ativar o item **Node Properties...**, selecionar o descritor criado automaticamente (*video1_Descriptor*, no exemplo) na lista **Descriptors**, e clicar no botão **Edit...** (Figura 41a). O Composer apresenta então o quadro de diálogo para edição do descritor (Figura 41b).

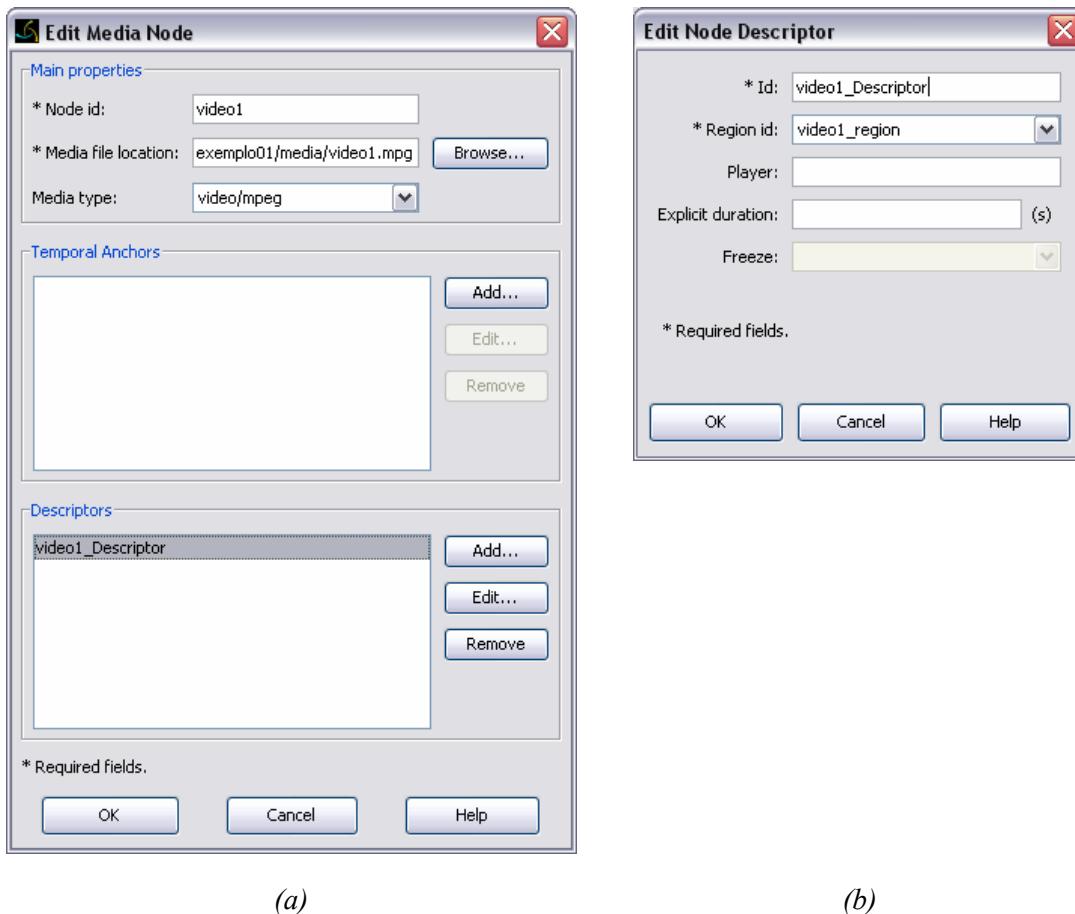


Figura 41. (a) Quadro de diálogo do nó apresentando o descritor criado automaticamente. (b) Quadro de diálogo para edição de atributos de um descritor.

Assim, observa-se que o Composer não impõe uma única estratégia para criação de documentos hipermídia, mas sim oferece flexibilidade para que o autor do documento adote a estratégia que lhe for mais conveniente no momento.

Sincronizando nós de mídia através de elos e conectores

Esta seção descreve dois exemplos de criação de elos para o sincronismo de mídias.

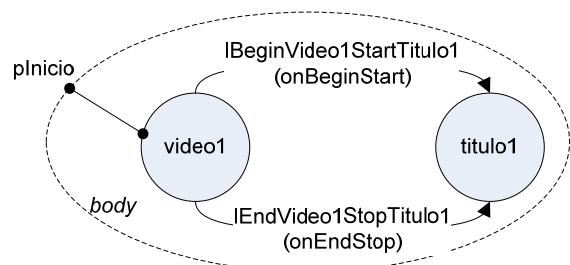
Exemplo 02 – Iniciando e terminando dois objetos de mídia simultaneamente

O objetivo deste exemplo é apresentar dois nós de mídia simultaneamente, um de texto (*titulo1*) e outro de vídeo (*video1*). Para isto, é necessário criar dois elos:

- elo *IBeginVideo1StartTitulo1*, para exibir *titulo1*, assim que *video1* iniciar, utilizando o conector *onBeginStart*
- elo *IEndVideo1StopTitulo1*, para terminar *titulo1*, assim que *video1* terminar, utilizando o conector *onEndStop*

Visões do documento

Visão Estrutural



Visão de leiaute

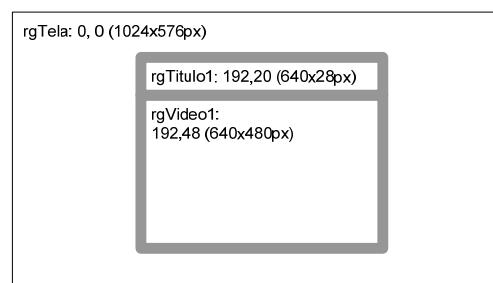


Figura 42. Visões estrutural e de leiaute do exemplo 02.

Visões Temporal e Espacial

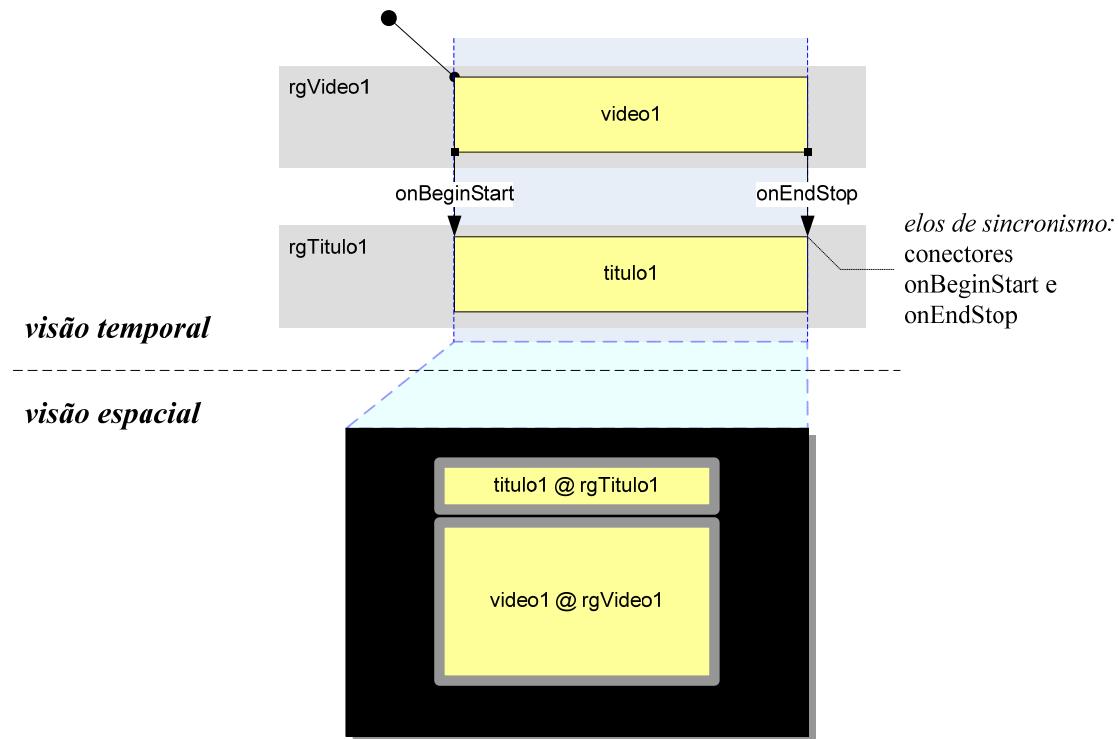


Figura 43. Visões temporal e espacial do exemplo 02, com sincronismo através dos conectores `onBeginStart` e `onEndStop`.

Passo-a-passo no Composer

Tomando como base o exemplo anterior, para construir um programa que sincronize o título e o vídeo, é necessário criar os seguintes elementos:

1. a região da tela onde o título será exibido, `rgTitulo1`;
2. o descritor que determina a forma como o título será exibido. Nesse caso, o texto associado a este descritor será exibido na região `rgTitulo1`, com os atributos *default* de exibição de texto;
3. o novo nó de mídia, o texto propriamente dito, `titulo1`;
4. o elo `lBeginVideo1StartTitulo1`, para iniciar a apresentação de `titulo1` assim que a apresentação de `video1` é iniciada; e
5. o elo `lEndVideo1StopTitulo1`, para terminar a apresentação de `titulo1` assim que a apresentação de `video1` é terminada.

Observações:

- Pode-se realizar os passos 1, 2 e 3 acima segundo a estratégia ilustrada no passo-a-passo alternativo, definindo primeiro o nó e depois a região, deixando que o Composer crie o descritor automaticamente.

- Para executar este exemplo, é necessário que haja um vídeo chamado `video1.mpg` e um arquivo de texto chamado `titulo1.txt`.
- A Listagem 2 do Apêndice I apresenta a listagem completa deste exemplo.

Passo 1: Criando região para o título

Como no exemplo anterior, deve-se definir, na visão de leiaute, uma nova região de tela. Acesse a visão de leiaute, selecione a região `rgTV` e accese o item de menu **Insert > Region...**, e defina a região `rgTitulo1` na posição (`left, top`) = (192, 20) e com dimensões (`width, height`) = (640, 28) (Figura 44).

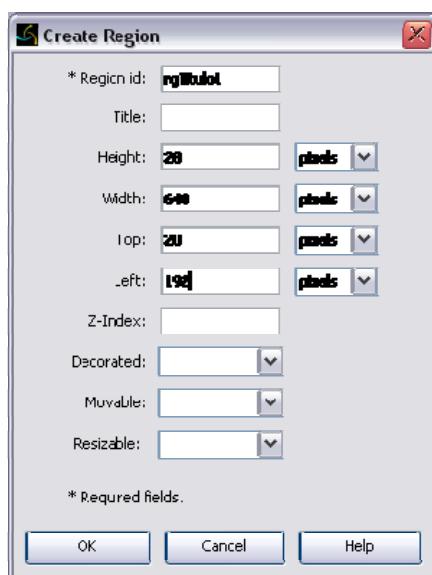


Figura 44. Região para exibir o título do vídeo `video1`.

A visão de leiaute deve apresentar as regiões `rgVideo1` e `rgTitulo1`, esta última em destaque, conforme a Figura 45:

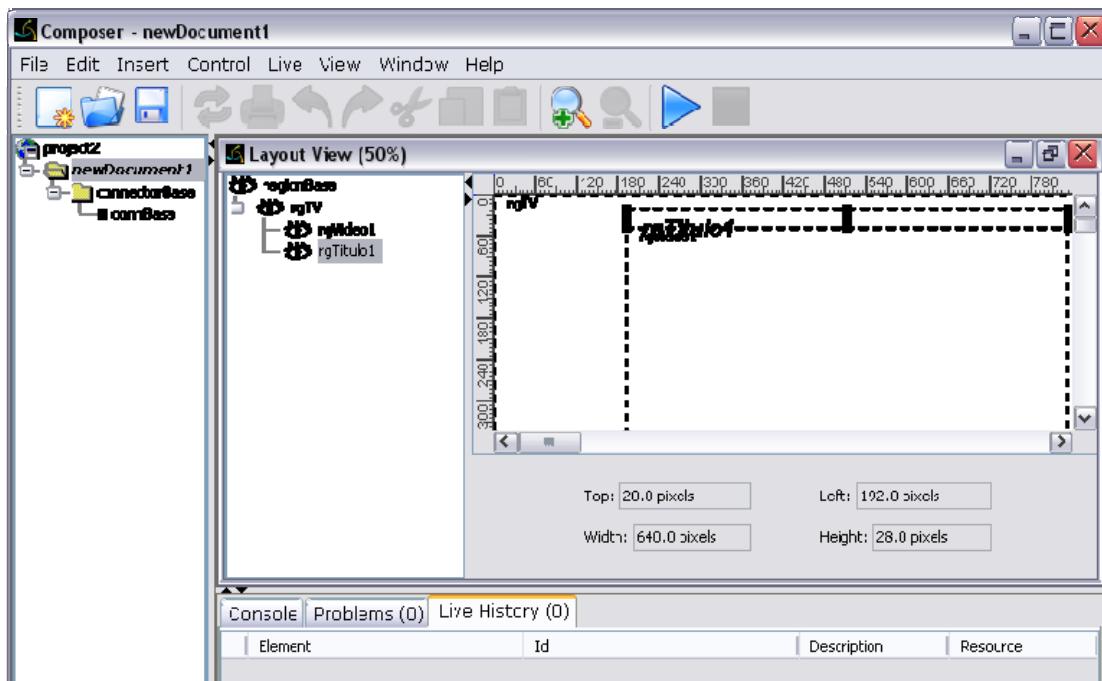


Figura 45. Visão de layout após a criação da região rgTitulo1.

Passo 2: Criando descriptor que determina como o título será exibido

Assim como no exemplo anterior, deve-se definir um descriptor para a apresentação do nó de mídia de título na região *rgTitulo1*. Acesse o item de menu **Insert > Descriptor...**, defina o **id** do descriptor como sendo *dVideo1*, **region** como *rgTitulo1* e confirme (Figura 46).



Figura 46. Quadro de diálogo para definição do descriptor dTitulo1.

Passo 3: Definindo a mídia correspondente ao título

Para criar o nó de mídia para o título, acesse a visão estrutural e o item de menu **Insert > Media Node....**. Defina o atributo **id** como sendo *titulo1* e **type** como *text/plain*; localize o arquivo que contém o texto do título; e associe o descriptor *dVideo1* ao nó (Figura 47).



Figura 47. Quadro de diálogo para criação de nó de texto *titulo1*.

A visão estrutural deve agora exibir o novo nó de mídia (Figura 48):

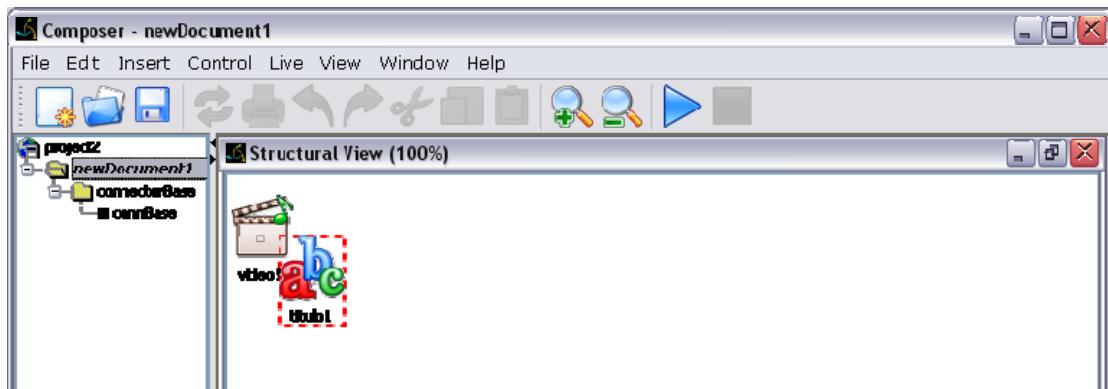


Figura 48. Visão estrutural após a criação do nó *titulo1*.

Para melhorar a visualização dos nós na visão estrutural, você pode mover um nó arrastando-o com o mouse.

Passos 4 e 5: Criando elos de sincronismo

O primeiro elo a ser criado é o elo *lBeginVideo1StartTitulo1*, que deve iniciar a apresentação do nó *titulo1* assim que a apresentação do nó *video1* é iniciada. Para isto, selecione os dois nós, clicando primeiro no nó *video1* e, pressionando a tecla Ctrl, clicando em seguida no nó

titulo1. Ambos os nós devem aparecer destacados com uma borda tracejada vermelha (Figura 49).

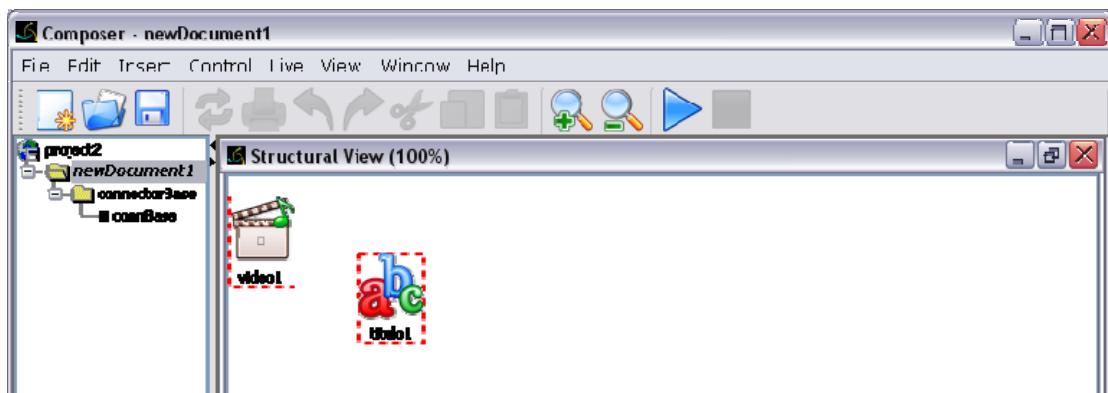


Figura 49. Visão estrutural com dois nós selecionados.

Acesse o item de menu **Insert > Causal link...** (Figura 50).

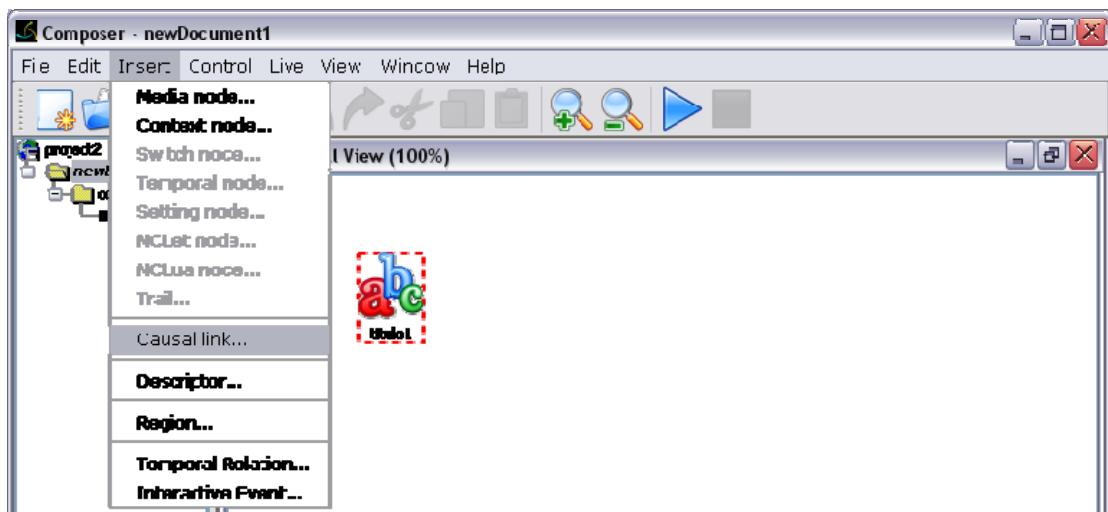


Figura 50. Item de menu **Insert > Causal link...**

O Composer apresenta um quadro de diálogo com os atributos e mapeamentos do elo (Figura 51):

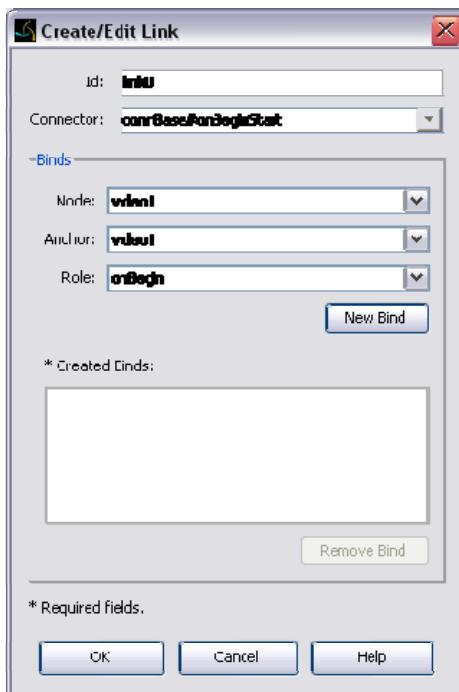


Figura 51. Quadro de diálogo para criação de um elo.

Para definir a semântica dos elos em NCL, utiliza-se conectores. Um conector define a semântica da relação entre o(s) nó(s) “de origem” do elo (i.e., que ativam o elo) e o(s) nó(s) “de destino” do elo. O Composer inclui, por default, um arquivo com conectores predefinidos, apresentado na combo **Connector**.

Neste exemplo, serão utilizados os conectores **onBeginStart** e **onEndStop**. A semântica dos conectores definidos atualmente será explicada adiante. Por ora, basta saber que o conector **onBeginStart** inicia a exibição de um ou mais nós de mídia (no papel *start*) assim que o nó de mídia “de origem” (no papel *onBegin*) é exibido, e o conector **onEndStop** termina a exibição de um ou mais nós de mídia (no papel *stop*) assim que o nó de mídia “de origem” (no papel *onEnd*) termina de ser exibido.

Para criar o primeiro elo, defina o **id** como *lBeginVideo1StartTitulo1*, selecione o **conector connBase#onBeginStart** e defina dois mapeamentos no grupo **Binds**:

a) associe o nó (**Node**) *video1* ao papel (**Role**) *onBegin*, clique em **New Bind**, e verifique se o mapeamento *video1/video1/onBegin* apareceu no quadro **Created Binds** (Figura 52a); e

b) associe o nó (**Node**) *titulo1* ao papel (**Role**) *start*, clique em **New Bind** e verifique se o mapeamento *titulo1/titulo1/start* apareceu no quadro **Created Binds** (Figura 52b):

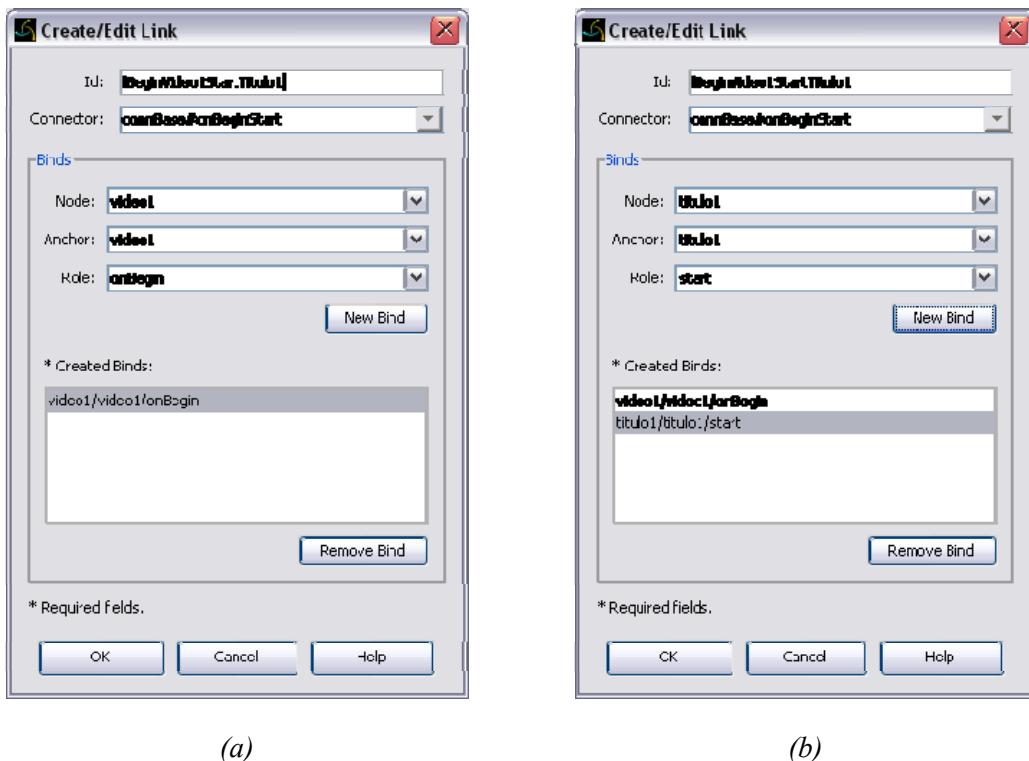


Figura 52. Quadros de diálogo para criação do elo lBeginVideo1StartTitulo1.

Clique em OK para confirmar a criação do elo e verifique, na visão estrutural, se o elo foi acrescentado entre os dois nós (Figura 53).

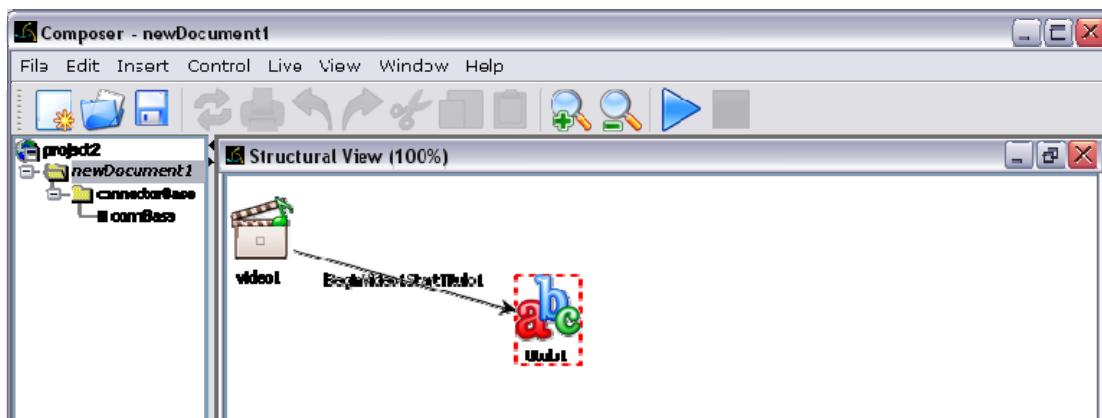


Figura 53. Visão estrutural exibindo elo lBeginVideo1StartTitulo1 entre os nós video1 e titulo1.

Em seguida, selecione novamente os dois nós, acesse o item de menu **Insert > Causal link...** e defina o elo *lEndVideo1StopTitulo1*, utilizando o conector *onEndStop* e mapeando o nó *video1* ao papel *onEnd* e o nó *titulo1* ao papel *stop* (Figura 54).

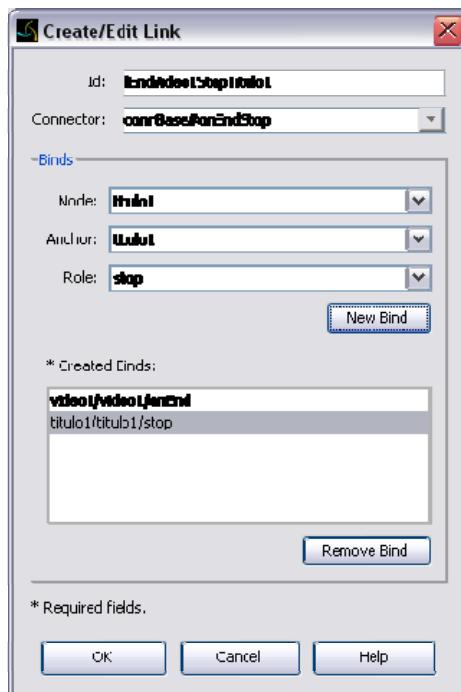


Figura 54. Quadro de diálogo para a criação do elo lEndVideo1StopTitulo1.

Como da outra vez, verifique, na visão estrutural, se o elo foi criado (Figura 55).

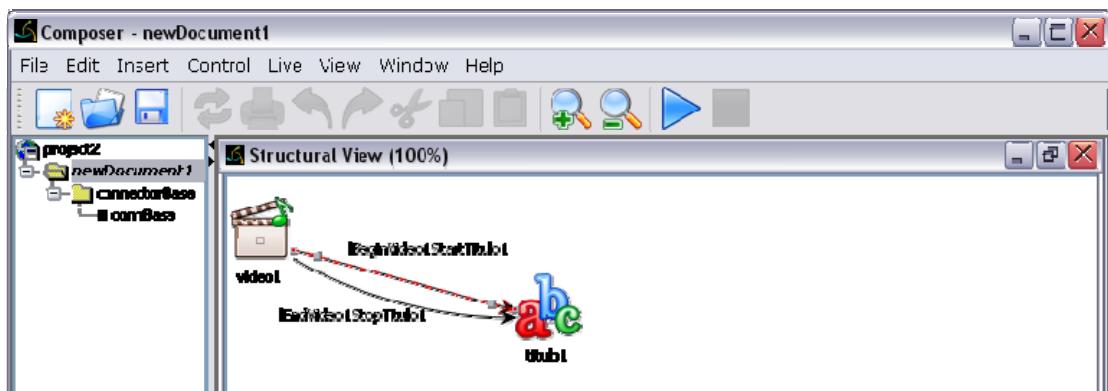


Figura 55. Visão estrutural apresentando os elos lBeginVideo1StartTitulo1 e lEndVideo1StopTitulo1.

Examinando a visão textual, observa-se o seguinte código NCL, correspondente às definições dos elos:

```
<link id="lBeginVideo1StartTitulo1" xconnector="connBase#onBeginStart">
    <bind component="video1" role="onBegin"/>
    <bind component="titulo1" role="start"/>
</link>

<link id="lEndVideo1StopTitulo1" xconnector="connBase#onEndStop">
    <bind component="video1" role="onEnd"/>
    <bind component="titulo1" role="stop"/>
</link>
```

Para saber mais: Elos

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 o 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 do seguinte elemento 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.

Para saber mais: Conectores

No NCM e na NCL, o sincronismo não é feito por marcas de tempo (*timestamps*), mas sim por mecanismos de causalidade e restrição definidos nos conectores (*connectors*). O conector define os papéis (*roles*) que os nós de origem e de destino exercem nos elos que utilizam o conector. A Figura 56 ilustra um conector ligando três nós.

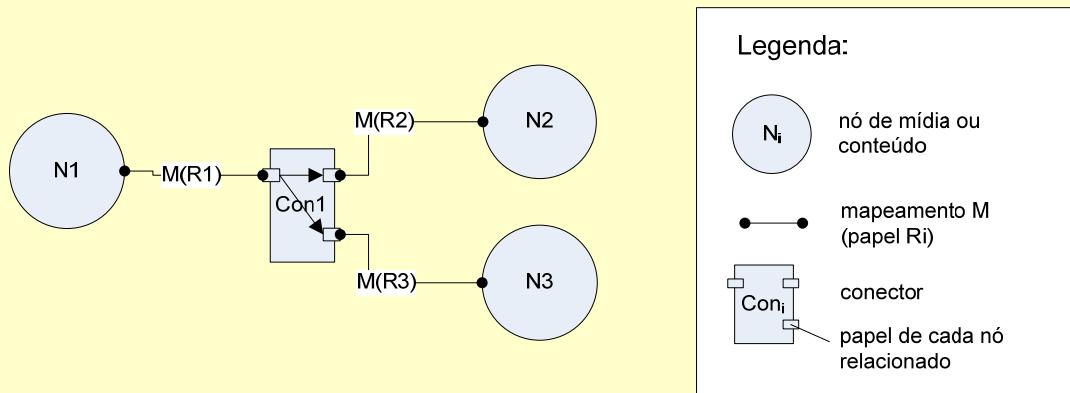


Figura 56. Ilustração de um conector ligando três nós.

Na NCL 3.0, existe apenas um tipo de conector: o conector causal (*causal connector*). Um conector causal 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 pelo 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.

Os seguintes papéis de **condição** são predefinidos:

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 for terminada (naturalmente ou por um evento 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 onResume for retomada (após uma pausa)
onSelection	... uma tecla <key> for pressionada, ou quando a tecla OK for pressionada enquanto um nó de mídia estiver com o foco
onAttribution	... um valor <value> for atribuído

Os seguintes papéis de **ação** são predefinidos:

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 pausa)

set ... estabelece um valor <value> à âncora associada ao papel **set**

Tanto os papéis de condição quanto os de ação estão associados a transições de estados numa máquina de eventos, ilustrada na Figura 57.:

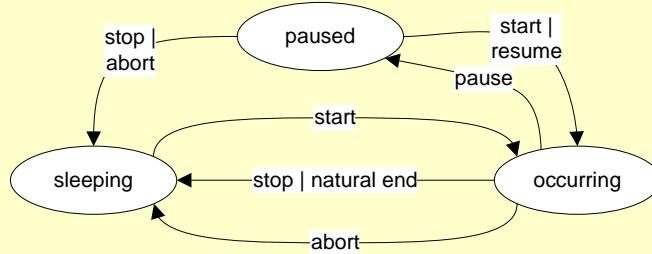


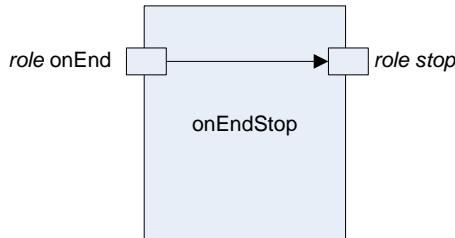
Figura 57. Máquina de estados de eventos.

As seguintes tabelas descrevem os conectores **onBeginStart** e **onEndStop** utilizados no exemplo 02:

Tabela 3. Definição do conector *onBeginStart*.

Nome:	onBeginStart
Condição:	início de exibição de um nó de mídia (mapeado no papel <i>onBegin</i>)
Ação:	dispara o início de exibição de um nó de mídia (mapeado no papel <i>start</i>)
Ilustração	<p>O diagrama mostra um conector com dois terminais. O terminal esquerdo é rotulado 'role onBegin' e o terminal direito 'role start'. Um bloco central contém a string 'onBeginStart'.</p>
Código NCL:	<pre> <causalConnector id="onBeginStart"> <simpleCondition role="onBegin"/> <simpleAction role="start"/> </causalConnector> </pre>
Leitura:	<p>Quando <onBegin> for iniciado, inicia <start>. (Substitui o papel entre <> pelo nó de mídia mapeado ao papel, no elo.)</p>

Tabela 4. Definição do conector *onEndStop*.

Nome:	onEndStop
Condição:	término de exibição de um nó de mídia (mapeado no papel <i>onEnd</i>)
Ação:	encerra a exibição de um nó de mídia (mapeado no papel <i>stop</i>)
Ilustração:	
Código NCL:	<pre><causalConnector id="onEndStop"> <simpleCondition role="onEnd"/> <simpleAction role="stop"/> </causalConnector></pre>
Leitura:	Quando <i><onEnd></i> terminar, termina <i><stop></i> .

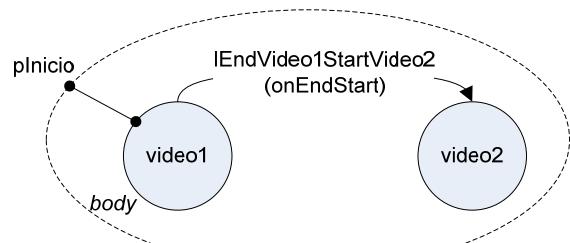
Esses conectores são bem simples, pois incluem apenas uma condição e uma ação. Mais adiante veremos outros conectores mais elaborados, que agrupam condições e ações, bem como permitem o mapeamento de mais de um nó a um mesmo papel.

Exemplo 03 – Iniciando um objeto de mídia quando outro termina

O objetivo deste exemplo é sincronizar o término de uma mídia com o início de outra, fazendo com que, ao término do vídeo *video1*, seja iniciada a exibição de um outro vídeo (*video2*), na mesma região. Além da criação do novo nó de mídia correspondente ao vídeo, será necessário também criar um elo para ligar os dois vídeos, através do conector **onEndStart**. Esse conector dispara a exibição de um nó de destino quando a exibição do nó de origem é terminada.

Visões do documento

Visão estrutural



Visão de leiaute

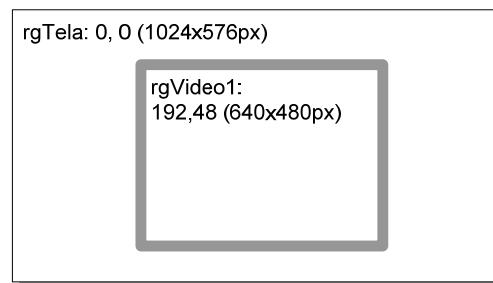


Figura 58. Visões estrutural e de leiaute do exemplo 03, onde o término de um vídeo dispara outro.

Visões temporal e espacial

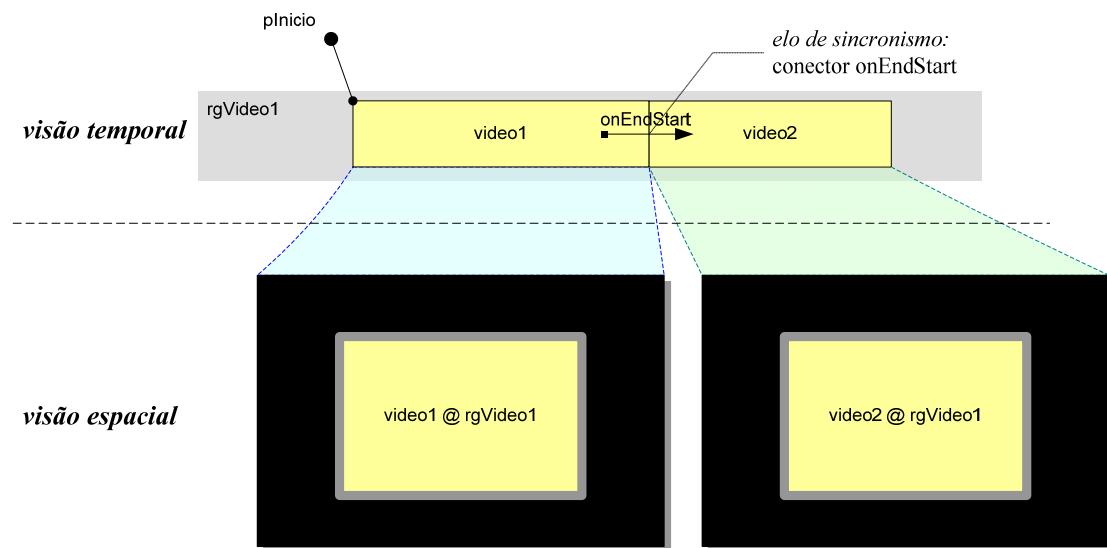


Figura 59. Visões temporal e espacial do exemplo 03.

Passo-a-passo no Composer

Tomando como base o exemplo 01, devem ser realizados os seguintes passos:

1. criação de um nó de mídia para o segundo vídeo, *video2*; e
2. criação de um elo *lEndVideo1StartVideo2*.

Observações:

- Para executar este exemplo, é necessário que haja dois vídeos, chamados *video1.mpg* e *video2.mpg*.
- A Listagem 3 do Apêndice I apresenta a listagem completa deste exemplo.

Passo 1: Definindo a mídia correspondente ao segundo vídeo

Para criar o nó de mídia para o título, acesse a visão estrutural, por meio do item de menu

Insert > Media Node.... Defina o atributo **id** como sendo *video2* e **type** como *video/mpeg*; localize o arquivo que contém o segundo vídeo; e associe o descritor *dVideo1* ao nó (Figura 60).

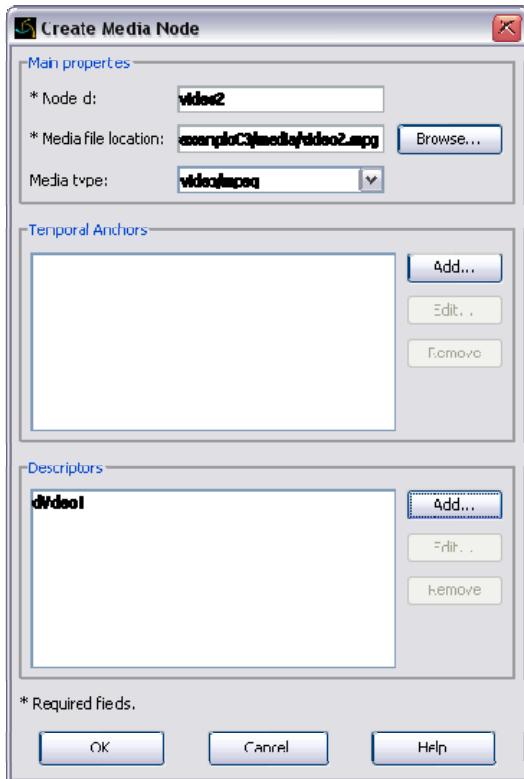


Figura 60. Criação do nó *video2*.

Verifique, na visão estrutural, se o nó foi criado (Figura 61).

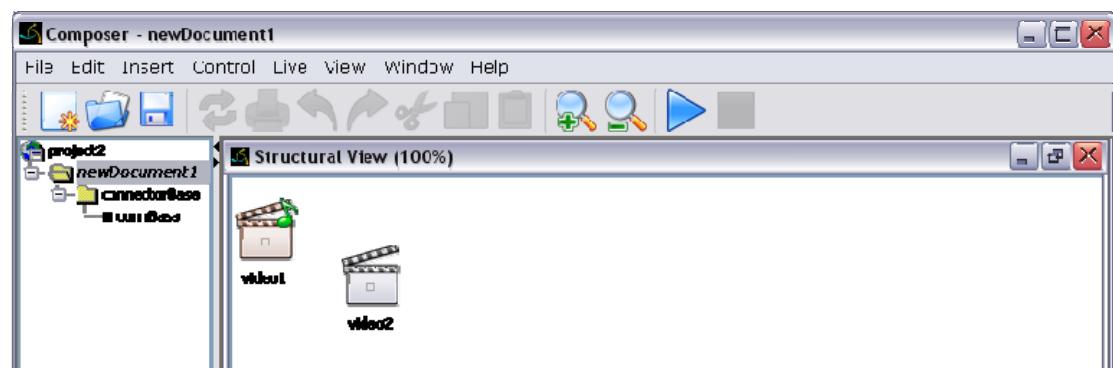


Figura 61. Visão estrutural apresentando os nós *video1* e *video2*.

Passo 2: Criando um elo de sincronismo

Neste exemplo, vamos criar um elo de sincronismo por manipulação direta. Sem que o nó *video1* esteja selecionado, pressione Ctrl e arraste o mouse do nó *video1* até o nó *video2*. Ao

soltar o botão do mouse, o Composer apresenta o quadro de diálogo de criação de elo, que deve ser preenchido da seguinte maneira: **id** como *lEndVideo1StartVideo2*, **connector** como *connBase#onEndStart*, e com os mapeamentos *video1* → *onEnd* e *video2* → *start* (Figura 62).

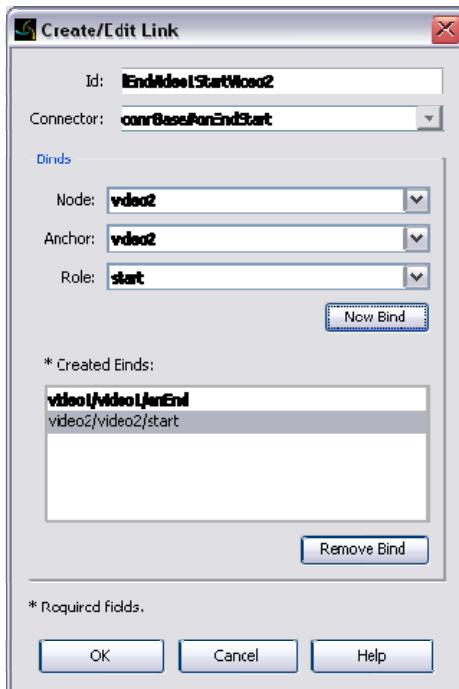


Figura 62. Criação do elo *lEndVideo1StartVideo2*.

Verifique, na visão estrutural, se o elo foi criado (Figura 63).

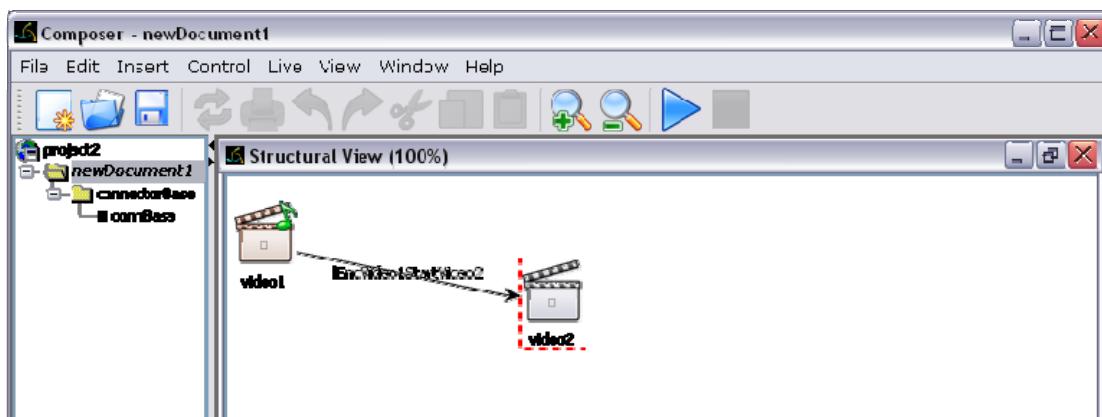
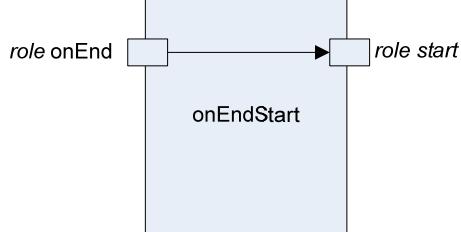


Figura 63. Visão estrutural apresentando o elo *lEndVideo1StartVideo2*.

Observa-se que, como o nó de vídeo deveria ser apresentado da mesma forma e na mesma região, não foi necessário criar um novo descritor nem uma nova região para o novo nó. Isto é uma evidência de que o custo de certos tipos de extensão a um programa existente pode ser extremamente baixo.

A tabela a seguir descreve o conector **onEndStart**, utilizado neste exemplo.

Tabela 5. Definição do conector *onEndStart*.

Nome:	onEndStart
Condição:	término de exibição de um nó de mídia (mapeado no papel <i>onEnd</i>)
Ação:	dispara o início de exibição de um nó de mídia (mapeado no papel <i>start</i>)
Ilustração:	
Código NCL:	<pre><causalConnector id="onEndStart"> <simpleCondition role="onEnd"/> <simpleAction role="start"/> </causalConnector></pre>
Leitura:	Quando <onEnd> terminar, inicia <start>.

Sincronizando diversos nós de mídia estática a um nó de mídia contínua

Exemplo 04 – Sincronizando um vídeo com diferentes arquivos de legenda

Esta seção apresenta um documento NCL que reproduz um vídeo no centro da tela e sincroniza uma legenda com o vídeo. A legenda é composta de três arquivos HTML, cada qual com um texto a ser sincronizado com um segmento do vídeo.

Como nos exemplos anteriores, deve-se criar os nós de mídia correspondentes ao vídeo e aos arquivos HTML. Para definir os pontos do vídeo em que a legenda deve aparecer, é necessário criar **âncoras** para o vídeo, cada qual definindo o intervalo de exibição da legenda correspondente. Em seguida, basta criar seis elos, cada qual para iniciar ou terminar a exibição de cada legenda. A origem de cada elo deve ser uma das âncoras do vídeo, e o destino a legenda correspondente.

Visões do documento

Visão estrutural

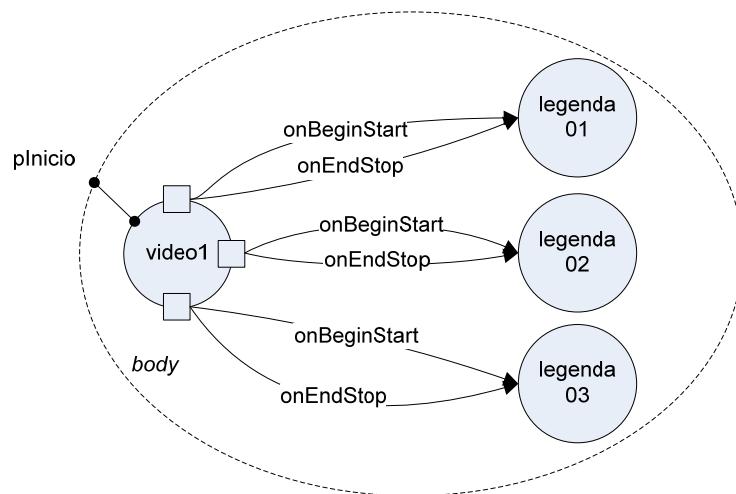


Figura 64. Visão estrutural do exemplo 04, de sincronização de legendas a segmentos de um vídeo.

Visão de leiaute

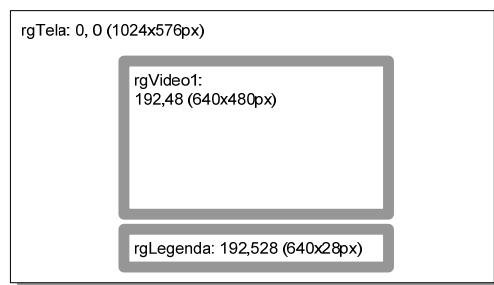


Figura 65. Visão de leiaute do exemplo 04, de sincronização de legendas a segmentos de um vídeo.

Visões temporal e espacial

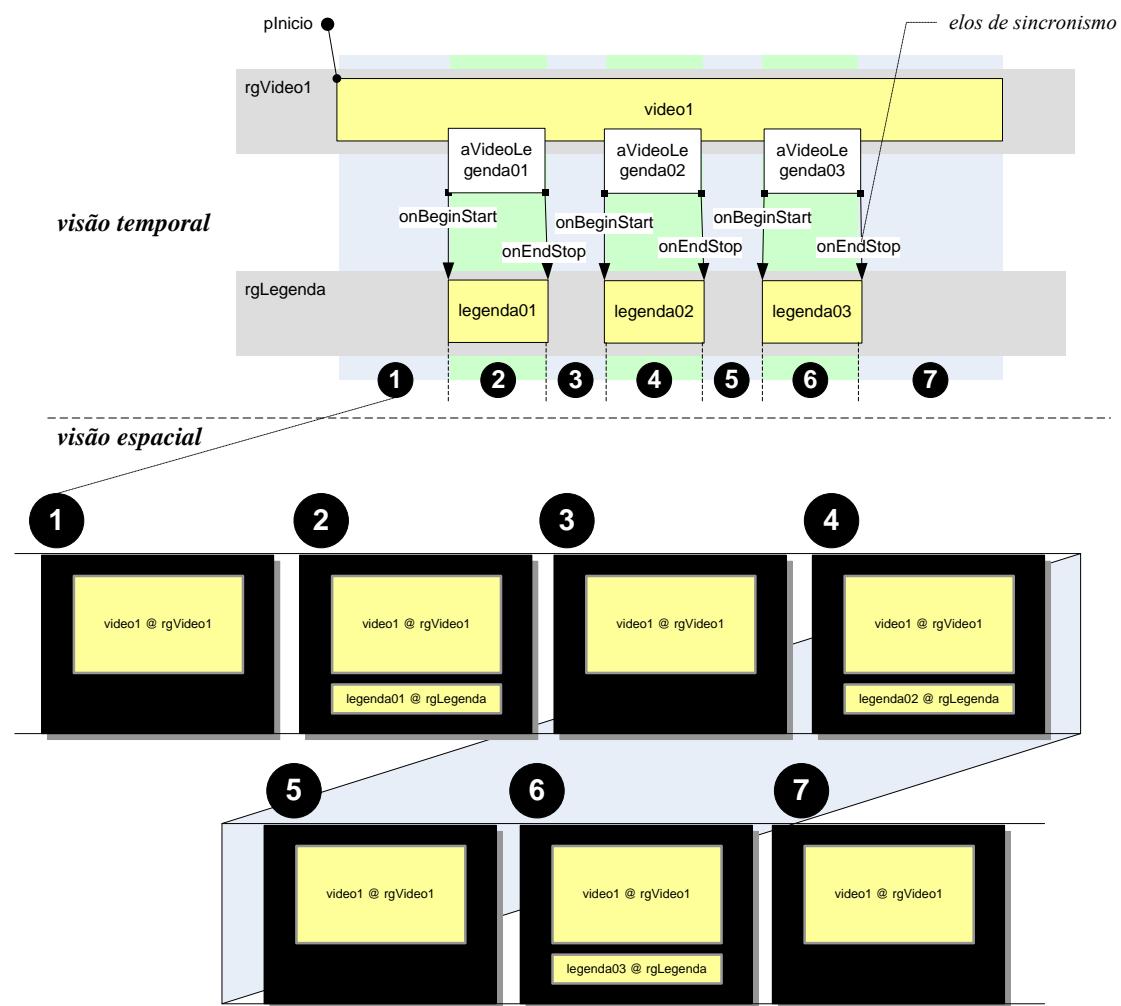


Figura 66. Visões temporal e espacial do exemplo 04.

Passo-a-passo no Composer

Tomando como base o exemplo 01, devem ser realizados os seguintes passos:

1. criação de uma região para a legenda, *rgLegenda*;

2. criação de um descritor para a legenda, *dLegenda*;
3. criação de três nós de mídia (*legenda1*, *legenda2* e *legenda3*), um para cada legenda;
4. criação das âncoras no vídeo que definem quando e por quanto tempo cada legenda deve ser apresentada (*aVideoLegenda01*, ...);
5. criação de elos de sincronização das legendas (*lLegenda01_start*, *lLegenda01_stop*, ...).

Observações:

- Para executar este exemplo, é necessário que haja um vídeo chamado video1.mpg e três arquivos HTML chamados legenda1.html, legenda2.html e legenda3.html.
- A Listagem 4 do Apêndice I apresenta a listagem completa deste exemplo.

Passo 1: Criando região para a legenda

Como nos exemplos anteriores, deve-se definir, na visão de leiaute, uma nova região de tela para a legenda. Acesse a visão de leiaute, selecione a região *rgTV* e acesse o item de menu **Insert > Region...**, e defina a região *rgLegenda* na posição (left, top) = (192, 528) e com dimensões (width, height) = (640, 28) (Figura 67).

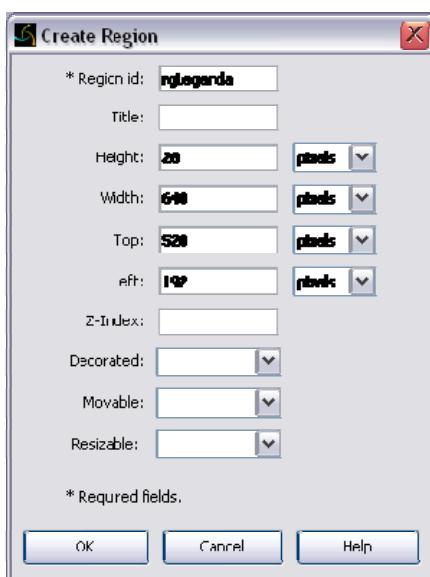


Figura 67. Criando região para a legenda.

Passo 2: Criando descritor para a legenda

Assim como nos exemplos anteriores, deve-se definir um descritor para a apresentação do nó de mídia de legenda na região *rgLegenda*. Acesse o item de menu **Insert > Descriptor...**, defina o **id** do descritor como sendo *dLegenda*, **region** como *rgLegenda* e confirme (Figura 68).



Figura 68. Criando descriptor para a legenda.

Passo 3: Definindo nós de texto HTML correspondentes às legendas

Para criar o primeiro nó de mídia para o título, acesse a visão estrutural e o item de menu **Insert > Media Node....**. Defina o atributo **id** como sendo *legenda1* e **type** como *text/html*; localize o arquivo que contém a primeira legenda; e associe o descriptor *dLegenda* ao nó (Figura 69). Repita este procedimento para criar os nós de mídia *legenda2* e *legenda3*.

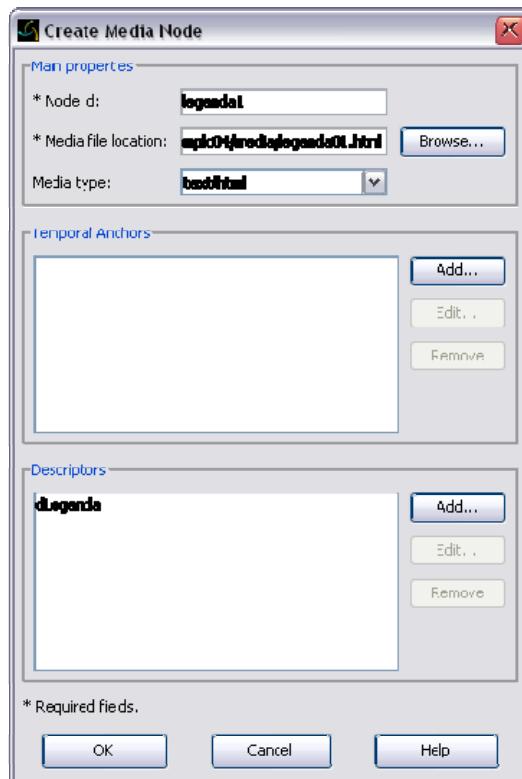


Figura 69. Criando um nó de texto para a legenda.

Na visão estrutural, verifique que todos os nós de legenda foram criados (Figura 70).

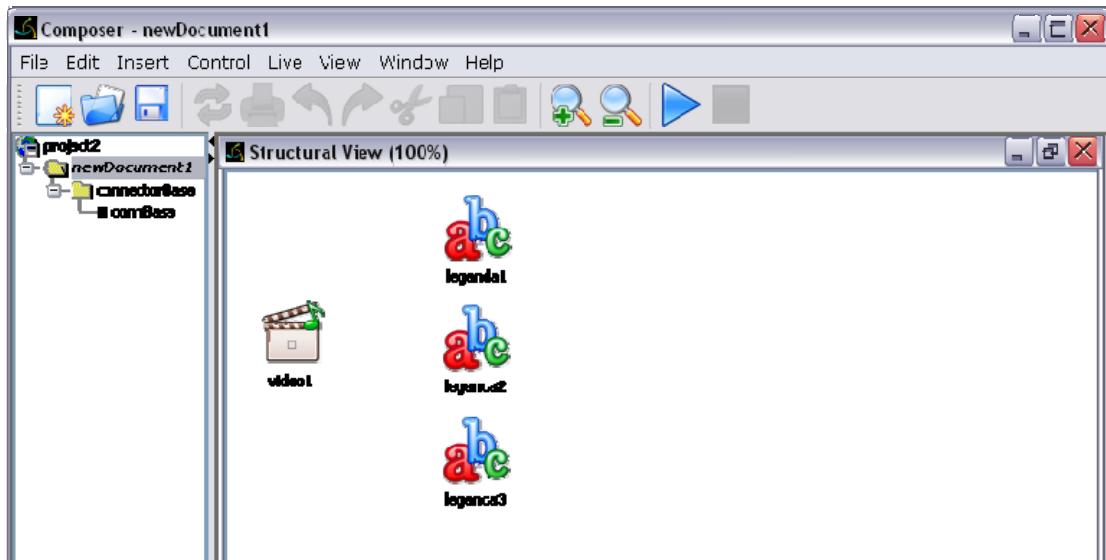


Figura 70. Visão estrutural apresentando um nó de vídeo e três nós HTML correspondentes a legendas.

Passo 4: Definindo âncoras no vídeo

Para definir onde cada legenda começa e termina de ser exibida, é necessário criar âncoras que segmentem o vídeo. Selecione o nó *video1* e acesse o item de menu **Edit > Node properties...**. No grupo **Temporal Anchors**, acrescente a âncora através do botão **Add....**. Defina o **id** da primeira âncora como sendo *aVideoLegenda01*, seu início (**Begin**) como 5 segundos e seu término (**End**) como 9 segundos (Figura 71).

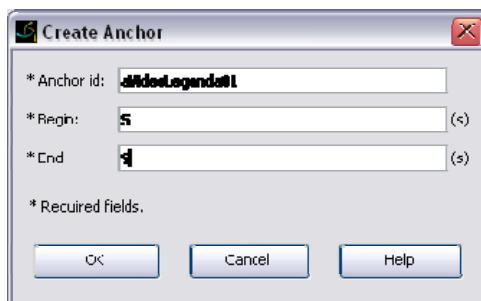


Figura 71. Criando âncora para a primeira legenda.

Clique em OK para confirmar a criação da âncora e repita a operação definindo mais duas âncoras, *aVideoLegenda02*, de 10 a 14 segundos, e *aVideoLegenda03*, de 15 a 19 segundos.



Figura 72. Nó video1 com as três âncoras definidas.

Passo 5: Criando elos de sincronismo para as legendas

Criar elos a partir de âncoras é semelhante a criar elos a partir de nós. A diferença é que, na hora do mapeamento, além de se especificar o nó, deve-se especificar também a âncora (atributo **interface** da linguagem NCL, que no Composer é chamado de **Anchor**) associada ao papel do conector. A Figura 73 ilustra a definição do elo *lLegenda01_start*, a) cujo papel *onBegin* está sendo mapeado para a âncora *aVideoLegenda01* do nó *video1*, e b) cujo papel *start* está sendo mapeado para o nó *legenda1*.

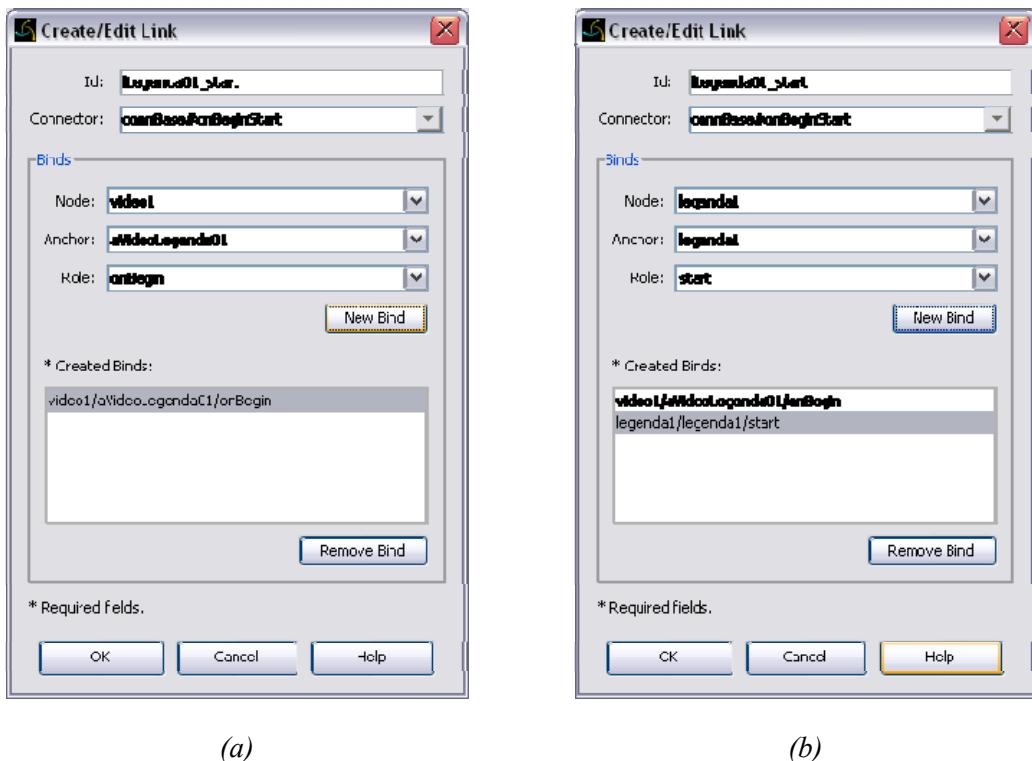


Figura 73. Criação de um elo que utiliza uma âncora em um de seus papéis.

O comportamento deste hiperdocumento é melhor compreendido na visão temporal (Figura 74).

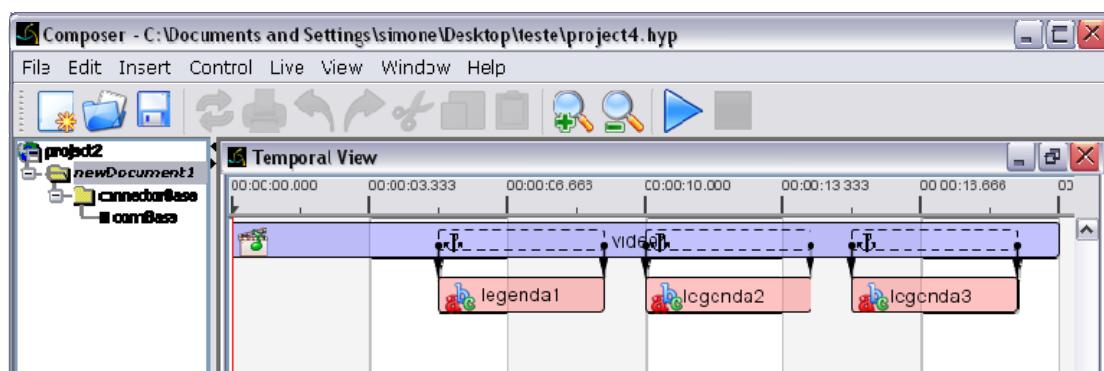


Figura 74. Visão temporal ilustrando o sincronismo entre âncoras de vídeo e as respectivas legendas.

Para saber mais: Âncoras

As âncoras são pontos de entrada para os nós de mídia ou contextos. O objetivo de se utilizar âncoras é utilizar segmentos ou propriedades de um nó de mídia ou contexto, seja como origem ou destino de elos. Uma âncora pode ser do tipo âncora de conteúdo (*content anchor*) ou âncora de propriedade (*property anchor*).

Uma **âncora de conteúdo** define um segmento da mídia (intervalo de tempo e/ou região no espaço) que poderá ser utilizado como ponto de ativação de elos. Um segmento de mídia é uma seleção contígua de **unidades de informação** (*information units*) de um nó. A definição dessas unidades de informação depende do tipo de mídia representado pelo nó. As unidades de informação de um vídeo, por exemplo, podem ser os *frames* do vídeo.

Uma âncora de conteúdo é definida como um elemento **area** dentro do elemento **media**. No exemplo abaixo, são definidas três âncoras de conteúdo para um nó de vídeo:

```
<media type="video" id="video1" src="media/video1.mpg" descriptor="dVideo1">
    <!-- âncoras de conteúdo no vídeo que devem ser sincronizadas com a legenda -->
    <area id="aVideoLegenda01" begin="5s" end="9s"/>
    <area id="aVideoLegenda02" begin="10s" end="14s"/>
    <area id="aVideoLegenda03" begin="15s" end="19s"/>
</media>
```

A NCL define os seguintes atributos de âncora de conteúdo:

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

Já as **âncoras de propriedade** se referem a propriedades de um nó de origem ou de

destino, que podem ser manipuladas pelos elos. Exemplos de propriedades são: volume de áudio de um nó de áudio ou vídeo, coordenadas e dimensões de exibição de um nó de mídia visual, entre outros.

Uma âncora de propriedade é definida como um elemento **property** dentro do elemento **media** ou **context**. No exemplo a seguir 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="dVideo1">
    <!-- âncoras de propriedade que serão manipuladas pelos elos -->
    <property name="top"/>
    <property name="left"/>
    <property name="width"/>
    <property name="height"/>

    <!-- âncora de conteúdo no vídeo que deve sincronizar a imagem -->
    <area id="aVideo1Imagem1" begin="3s" end="8s"/>
</media>
```

Exemplo 05 – Sincronizando um vídeo com um único arquivo de legenda, segmentado

Esta seção também apresenta um documento NCL que reproduz um vídeo no centro da tela e sincroniza uma legenda com o vídeo. Desta vez, no entanto, a legenda é composta de um único arquivo HTML, contendo três elementos DIV, cada qual com um texto a ser sincronizado com o vídeo, como no seguinte exemplo:

```
<html>
<body>
    <div id="legenda01">Legenda 1</div>
    <div id="legenda02">Legenda 2</div>
    <div id="legenda03">Legenda 3</div>
</body>
</html>
```

Assim como no exemplo anterior, para definir os pontos do vídeo em que a legenda deve aparecer, é necessário criar âncoras para o vídeo, cada qual definindo o intervalo de exibição da legenda correspondente. Em seguida, basta criar seis elos, cada qual para iniciar ou terminar a exibição de cada legenda. A origem de cada elo deve ser uma das âncoras do vídeo, e o destino a legenda correspondente.

Diferentemente do exemplo anterior, em vez de os três nós de legenda constituírem arquivos distintos, sua fonte (atributo **src**) ficará no formato **nome_do_arquivo#id_do_div**, como em *legendas.html#legenda01* (Figura 75).

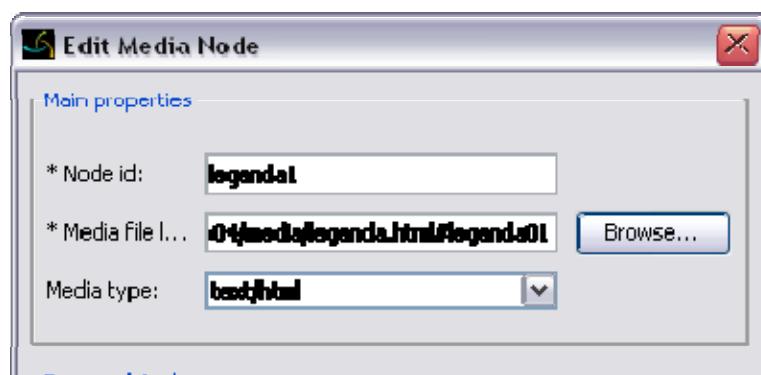


Figura 75. Criação de nó com fragmento de arquivo.

As visões temporal e espacial deste exemplo são as mesmas do exemplo anterior, ilustradas na Figura 66.

Interação com o Usuário

Esta seção descreve um exemplo de criação de elo para a interação do usuário com o documento hipermídia.

Exemplo 06 – Exibindo um vídeo em loop até a intervenção do usuário

O objetivo deste exemplo é exibir um vídeo em *loop*, permitindo ao usuário prosseguir para um segundo vídeo, pressionando a tecla verde do controle remoto da TV digital. Ambos os vídeos devem ser apresentados na mesma posição da tela.

Para que um vídeo possa ser exibido em *loop*, pode-se utilizar o conector **OnEndStart** com o mesmo nó de vídeo nos papéis **onEnd** e **start**. Para disparar o segundo vídeo, no entanto, não basta utilizar o evento **stop**, pois este dispararia o primeiro elo novamente (iniciando-o mais uma vez, mantendo o *loop*). Portanto, é necessário que o vídeo em *loop* seja interrompido por um evento **abort**. Essa relação foi definida em um novo conector chamado **OnKeySelectionStartNStopNAbortN**.

Visões do documento

Visão estrutural

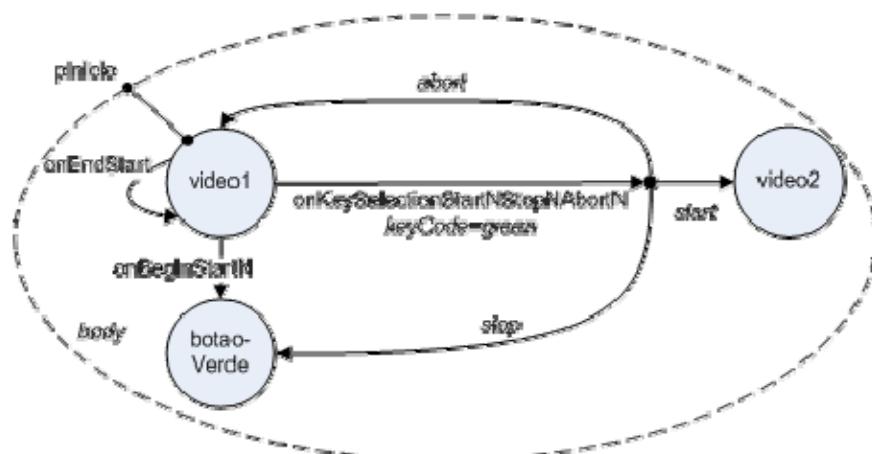


Figura 76. Visão estrutural do exemplo 06, com sincronismo (conectores **onBeginStart** e **onEndStart**) e interatividade (conector **onKeySelectionStartNStopNAbortN**).

Visão de leiaute

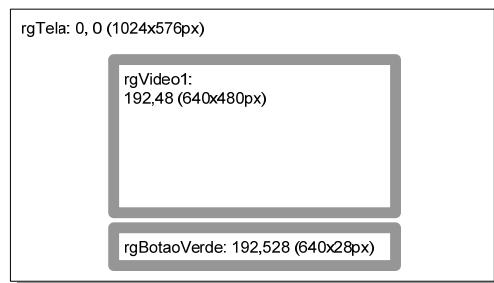


Figura 77. Visão de leiaute do exemplo 06.

Visões temporal e espacial

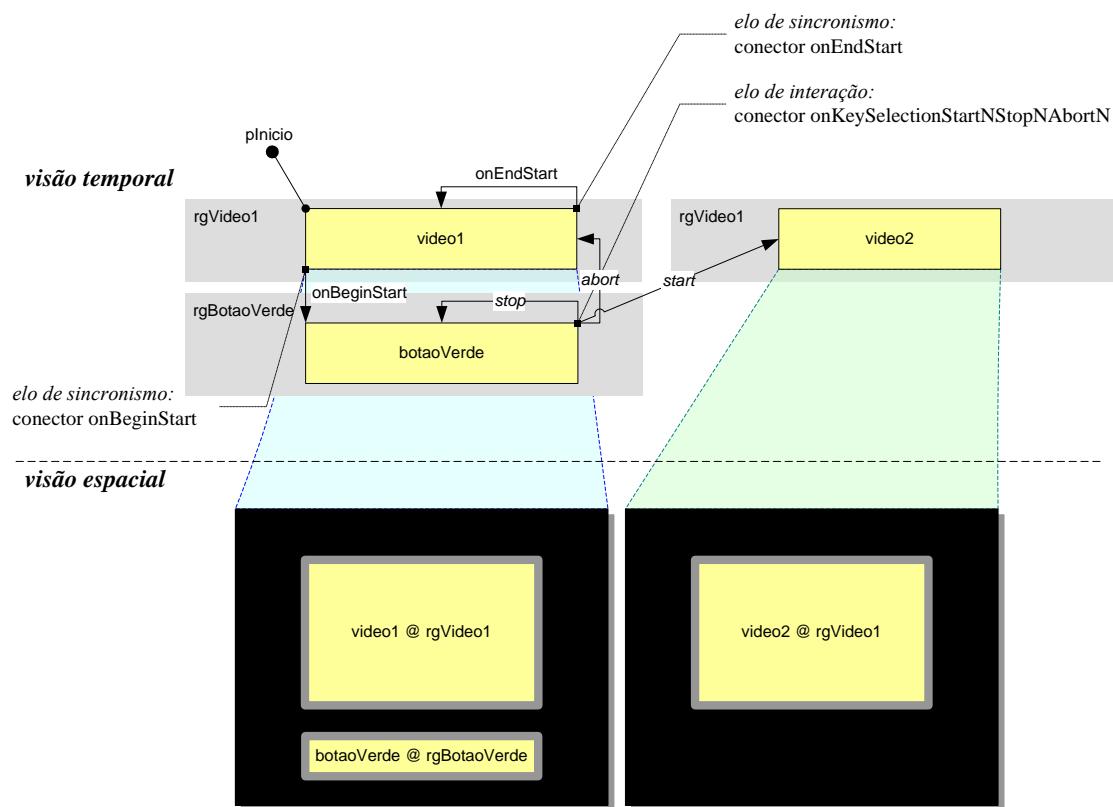


Figura 78. Visões temporal e espacial do exemplo 06.

Atualmente, não se pode criar novos conectores com apoio da ferramenta Composer. Sendo, assim, torna-se necessário utilizar a visão textual ou mesmo editar o código NCL em um editor de texto. Existem duas alternativas: a) criar um novo arquivo de conectores e importá-lo na seção **connectorBase**; ou b) definir os novos conectores no próprio hiperdocumento.

Alternativa a: Novo arquivo de conectores

arquivo exemplo06a.ncl

```

1:   <connectorBase>
2:     <importBase alias="connBase" documentURI="maestroConnectorBase.conn"/>
3:     <importBase alias="meusConectores" documentURI="meusConectores.conn"/>
4:   </connectorBase>

```

```

5: ...
6:     <link id="ISelectBotaoVerde" xconnector="meusConectores#onKeySelectionStartNStopNAbrtN">

```

arquivo meusConectores.conn

```

1:   <?xml version="1.0" encoding="ISO-8859-1"?>
2:
3:   <ncl id="meusConectores" xmlns="http://www.ncl.org.br/NCL3.0/CausalConnectorProfile"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.ncl.org.br/NCL3.0/CausalConnectorProfile
      http://www.ncl.org.br/NCL3.0/profiles/NCL30CausalConnector.xsd">
4:
5:   <head>
6:     <connectorBase>
7:
8:       <causalConnector id="onKeySelectionStartNStopNAbrtN">
9:         <connectorParam name="keyCode" />
10:        <simpleCondition role="onSelection" key="$keyCode"/>
11:        <compoundAction operator="par">
12:          <simpleAction role="start" max="unbounded" qualifier="par"/>
13:          <simpleAction role="stop" max="unbounded" qualifier="par"/>
14:          <simpleAction role="abort" max="unbounded" qualifier="par"/>
15:        </compoundAction>
16:      </causalConnector>
17:
18:    </connectorBase>
19:  </head>
20: </ncl>

```

Observação: Ao se importar mais do que um arquivo de conectores, é necessário se certificar que cada arquivo tem um **id** diferente (no caso, *meusConectores*, linha 3).

Alternativa b: Definição de novo conector embutida no documento NCL

arquivo exemplo06b.ncl

```

1:   <connectorBase>
2:     <importBase alias="connBase" documentURI="maestroConnectorBase.conn"/>
3:
4:     <causalConnector id="onKeySelectionStartNStopNAbrtN">
5:       <connectorParam name="keyCode" />
6:       <simpleCondition role="onSelection" key="$keyCode"/>
7:       <compoundAction operator="par">
8:         <simpleAction role="start" max="unbounded" qualifier="par"/>
9:         <simpleAction role="stop" max="unbounded" qualifier="par"/>
10:        <simpleAction role="abort" max="unbounded" qualifier="par"/>
11:      </compoundAction>
12:    </causalConnector>
13:
14:  </connectorBase>
15: ...
16:  <link id="ISelectBotaoVerde" xconnector="onKeySelectionStartNStopNAbrtN">
17:

```

A tabela a seguir descreve o conector **onKeySelectionStartNStopNAbrtN**. Observa-se que esse conector exige que o elo correspondente defina um parâmetro do tipo:

- **bindParam** ou **linkParam**, para preencher o valor do parâmetro *keyCode*, definido no conector como a tecla de acionamento do elo.

Tabela 6. Conector *onKeySelectionStartNStopNAbortN*.

Nome:	onKeySelectionStartNStopNAbortN
Condição:	tecla <keyCode> acionada (papel <i>onSelection</i>)
Ação:	<ul style="list-style-type: none"> ▪ exibe as mídias identificadas pelo papel <i>start</i>; ▪ pára as mídias identificadas pelo papel <i>stop</i>; ▪ aborta as mídias identificadas pelo papel <i>abort</i>.
Código NCL:	<pre><causalConnector id="onKeySelectionStartNStopNAbortN"> <connectorParam name="keyCode" /> <simpleCondition role="onSelection" key="\$keyCode"/> <compoundAction operators="par"> <simpleAction role="start" max="unbounded" qualifier="par"/> <simpleAction role="stop" max="unbounded" qualifier="par"/> <simpleAction role="abort" max="unbounded" qualifier="par"/> </compoundAction> </causalConnector></pre>
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 tecla do controle remoto associada à seleção, através do parâmetro keyCode . Por 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, 0
- MENU
- e outros dependentes do dispositivo de entrada: INTERACTION, *, #, letras A-Z

Redimensionando regiões

Exemplo 07 – Redimensionando uma região de vídeo durante a exibição de uma imagem

Esta seção apresenta um documento NCL que reproduz um vídeo em tela inteira (1024 x 576 px) e que, num determinado momento, é redimensionado para dar lugar a uma imagem no quadrante inferior direito da tela.

Nesse exemplo, vamos redimensionar o vídeo quando uma imagem aparecer, a 3 segundos do início do vídeo, e restaurar o tamanho original, a 8 segundos do vídeo, quando a imagem deve ser ocultada.

Visões do documento

Visão estrutural

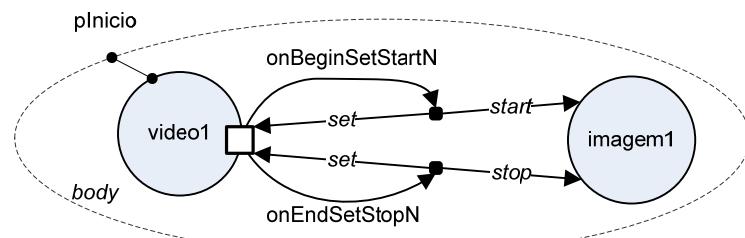


Figura 79. Visão estrutural do exemplo 07, documento que redimensiona um vídeo enquanto uma imagem é exibida.

Visão de leiaute

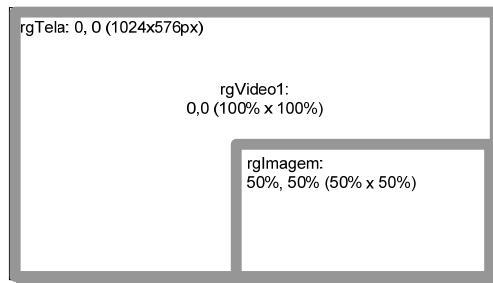


Figura 80. Visão de leiaute do exemplo 07.

Visões temporal e espacial

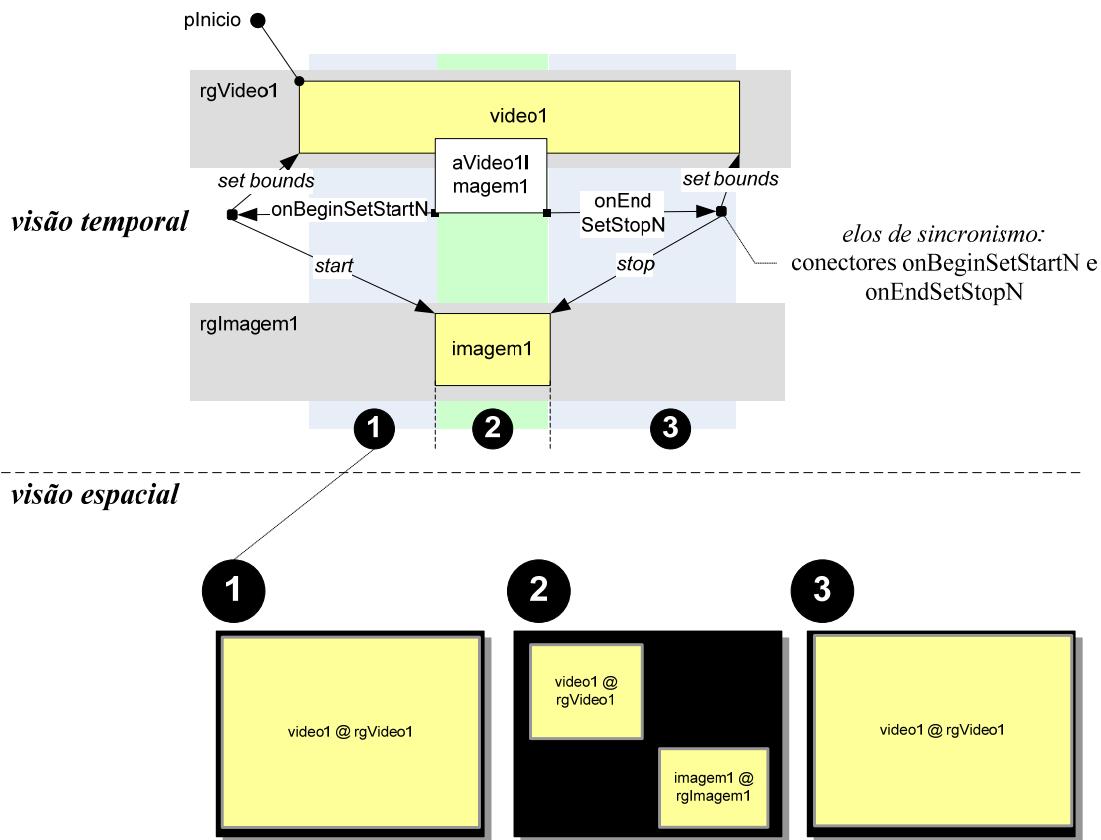


Figura 81. Visões temporal e espacial do exemplo 07.

Para manipular as propriedades **top**, **left**, **width** e **height** da região à qual o vídeo está associado, é necessário definir explicitamente essas propriedades como âncoras da mídia. A versão atual do Composer não permite definir âncoras de propriedade.

No caso específico das propriedades desse exemplo, pode-se especificar uma âncora para a propriedade **bounds** para manipulá-las em conjunto, na ordem **left**, **top**, **width**, **height**:

```
<media type="video/mpeg" id="video1" src="media/video1.mpg" descriptor="dVideo1">
  <!-- âncora no vídeo que deve sincronizar a imagem -->
  <area id="aVideo1Imagem1" begin="3s" end="8s"/>
  <!-- propriedade que será manipulada pelos elos -->
  <property name="bounds" />
</media>
```

A manipulação é feita através dos elos *Video1Imagem1_start* e *Video1Imagem1_stop*:

```
<!-- âncora do vídeo 1 deve iniciar a exibição da imagem -->
<link id="Video1Imagem1_start" xconnector="meusConectores#onBeginSetStartN">
  <bind component="video1" interface="aVideo1Imagem1" role="onBegin" />
  <bind component="video1" interface="bounds" role="set">
    <bindParam name="bounds" value="0,0,50%,50%" />
  </bind>
  <bind component="imagem1" role="start" />
</link>
```

```
<!-- âncora do vídeo 1 deve terminar a exibição da imagem -->
<link id="Video1Imagem1_stop" xconnector="meusConectores#onEndSetStopN">
  <bind component="video1" interface="aVideo1Imagem1" role="onEnd" />
  <bind component="video1" interface="bounds" role="set">
    <bindParam name="bounds" value="0,0,100%,100%" />
  </bind>
  <bind component="imagem1" role="stop" />
</link>
```

Os conectores **onBeginSetStartN** e **onEndSetStopN** são definidos nas tabelas a seguir:

Tabela 7. Conector onBeginSetStartN.

Nome:	onBeginSetStartN
Condição:	início de exibição da(s) mídia(s) no papel <i>onBegin</i>
Ação:	<ul style="list-style-type: none"> ▪ 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>; ▪ 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). Por exemplo:

```
<bind component="video1" interface="bounds" role="set">
  <bindParam name="var" value="0,0,100%,100%" />
</bind>
```

A tabela a seguir descreve o conector **onEndSetStopN**.

Tabela 8. Conector onEndSetStopN.

Nome:	onEndSetStopN
Condição:	término de exibição da mídia no papel <i>onEnd</i>
Ação:	<ul style="list-style-type: none"> ▪ 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 da(s) mídia(s) no papel <i>stop</i>.

Código
NCL:

```
<causalConnector id="onEndSetStopN">
  <connectorParam name="bounds"/>
  <simpleCondition role="onEnd"/>
  <compoundAction operator="seq">
    <simpleAction role="set" value="$bounds"/>
    <simpleAction role="stop" max="unbounded" qualifer="par"/>
  </compoundAction>
</causalConnector>
```

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).

Exemplo 08 – Alternando entre mídias em resposta a ações do usuário

O objetivo deste exemplo é permitir ao usuário alternar entre dois vídeos, através da seleção das teclas vermelha e verde do controle remoto da TV digital, como em uma mudança de câmera em uma partida de futebol, por exemplo. Ambos os vídeos devem ser apresentados na mesma posição da tela. Para alternar entre os vídeos, algumas modificações precisam ser feitas com relação ao Exemplo 03.

Os dois vídeos devem ser iniciados de forma sincronizada, sendo que um deles deve iniciar invisível e sem som. Quando o usuário fizer uma seleção com os botões do controle remoto, os valores das propriedades dos dois vídeos devem ser invertidos.

Nesse exemplo, os dois botões ficarão visíveis o tempo todo. Isso é uma decisão ruim de projeto, pois acionar um dos botões não terá efeito nenhum. No exemplo 09 resolveremos este problema: somente o botão que trocar o vídeo atual será exibido.

Visões do documento

Visão estrutural

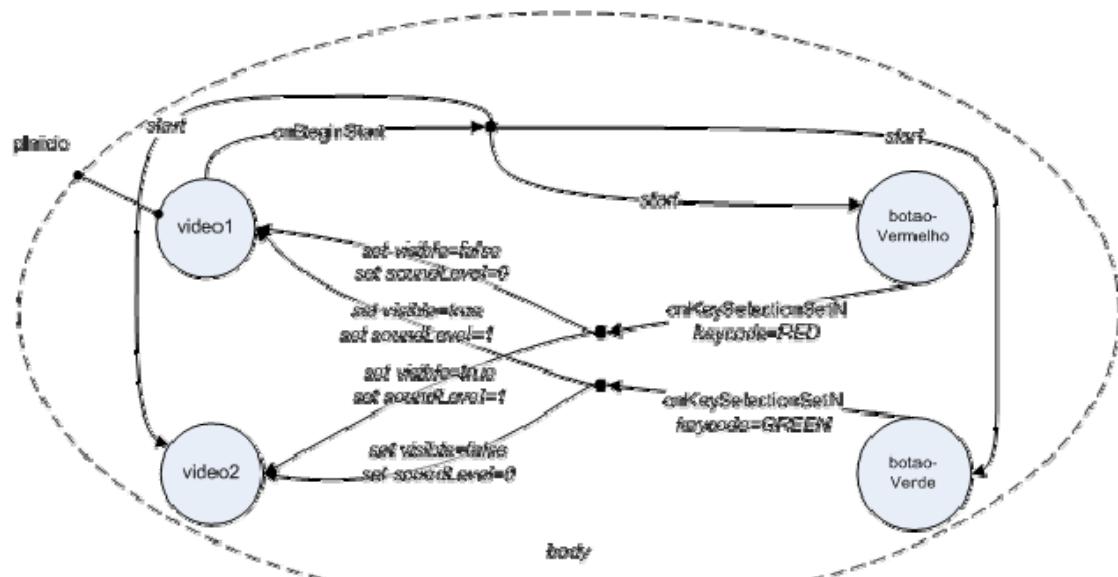


Figura 82. Visão estrutural do exemplo 08.

Visão de leiaute



Figura 83. Visão de leiaute do exemplo 08.

Visões temporal e espacial

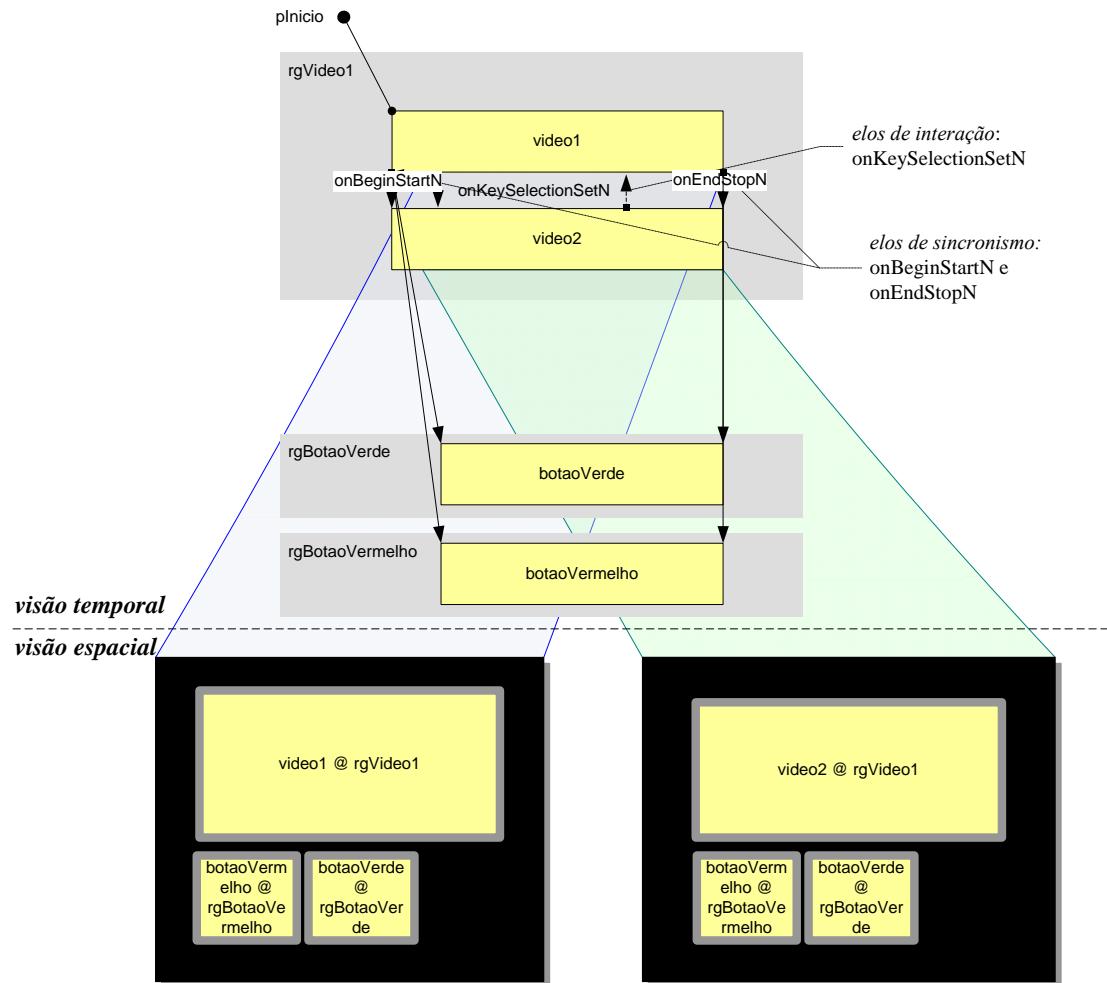


Figura 84. Visões temporal e espacial do exemplo 08.

Como as duas mídias serão iniciadas ao mesmo tempo mas com valores de propriedades de visibilidade e som distintos, é necessário criar um novo descritor para o segundo vídeo, reusando a mesma região do *video1* mas com os valores das propriedades correspondentes às definidas no seu descritor para que não haja som nem exibição no início:

```
<descriptor id="dVideo2" region="rgVideo1">
  <descriptorParam name="visible" value="false" />
  <descriptorParam name="soundLevel" value="0"/>
</descriptor>
```

Para que seja possível alterar as propriedades **visible** e **soundLevel**, é necessário criar âncoras de propriedades nos nós de mídia correspondentes:

```
<media type="video/mpeg" id="video1" src="media/video1.mpg" descriptor="dVideo1">
  <property name="visible" />
  <property name="soundLevel" />
</media>

<media type="video/mpeg" id="video2" src="media/video2.mpg" descriptor="dVideo2">
  <property name="visible" />
```

```
<property name="soundLevel" />
</media>
```

Finalmente, é necessário criar dois elos para efetuar a seleção do vídeo, através do conector **onKeySelectionSetN**. Como será visto adiante, esse conector é utilizado para trocar as propriedades de visibilidade e volume do som, trocando o vídeo sendo exibido conforme o botão selecionado: o botão vermelho ativa o *video2* e o botão verde ativa o *video1*:

```
<!-- alterna para vídeo 2 caso a tecla vermelha seja acionada -->

<link id="ISelecionaVideo2" xconnector="connBase#onKeySelectionSetN">
  <bind component="botaoVermelho" role="onSelection">
    <bindParam name="keyCode" value="RED"/>
  </bind>
  <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>
  <bind component="video2" interface="visible" role="set">
    <bindParam name="var" value="true"/>
  </bind>
  <bind component="video2" interface="soundLevel" role="set">
    <bindParam name="var" value="1"/>
  </bind>
</link>

<!-- alterna para vídeo 1 caso a tecla verde seja acionada -->

<link id="ISelecionaVideo1" xconnector="connBase#onKeySelectionSetN">
  <bind component="botaoVerde" role="onSelection">
    <bindParam name="keyCode" value="GREEN"/>
  </bind>
  <bind component="video2" interface="visible" role="set">
    <bindParam name="var" value="false"/>
  </bind>
  <bind component="video2" interface="soundLevel" role="set">
    <bindParam name="var" value="0"/>
  </bind>
  <bind component="video1" interface="visible" role="set">
    <bindParam name="var" value="true"/>
  </bind>
  <bind component="video1" interface="soundLevel" role="set">
    <bindParam name="var" value="1"/>
  </bind>
</link>
```

Observa-se que, como o elo deve alterar algumas propriedades dos nós de vídeo, cada elemento de ligação (**bind**) deve especificar não apenas o componente de destino do elo (atributo **component**), mas também a sua âncora de propriedade (através do atributo **interface**), tal como definida no nó da mídia. Além disso, como o valor dessa propriedade deve ser modificado, o novo valor deve ser passado como parâmetro (através do elemento **bindParam**):

```
<bind component="video1" interface="aVideo1Visible" role="set">
  <bindParam name="var" value="false" />
</bind>
```

```
<bind component="video1" interface="aVideo1SoundLevel" role="set">
    <bindParam name="var" value="0" />
</bind>
```

A tabela a seguir descreve o conector **onKeySelectionSetN**. Observa-se que esse conector exige que o elo correspondente defina um parâmetro (**linkParam** ou **bindParam**), para preencher o valor da propriedade *keyCode*, que identifica a tecla pressionada.

Tabela 9. Conector onKeySelectionSetN.

Nome:	onKeySelectionSetN
Condição:	tecla <keyCode> acionada (papel <i>onSelection</i>)
Ação:	altera propriedades dos nós associados ao papel <i>set</i> , através do parâmetro <i>var</i>
Código NCL:	<pre><causalConnector id="onKeySelectionSetN"> <connectorParam name="keyCode" /> <connectorParam name="var"/> <simpleCondition role="onSelection" key="\$keyCode" /> <simpleAction role="set" value="\$var" max="unbounded" qualifier="par"/> </causalConnector></pre>
Observações:	<p>Os elos que utilizarem este conector deverão identificar, como parâmetro adicional do elo (linkParam), ou como parâmetro do mapeamento de um nó (bindParam), o código virtual da tecla do controle remoto associada à seleção. Por exemplo:</p> <pre><bindParam name="keyCode" value="VK_ENTER"/></pre> <p>Os elos que utilizarem este conector deverão identificar também, como parâmetro adicional dos mapeamentos dos nós (bindParam), os novos valores das propriedades. No exemplo, as propriedades de visibilidade e volume de som, como a seguir:</p> <pre><bind component="video1" interface="aVideo1Visible" role="set"> <bindParam name="var" value="false" /> </bind> <bind component="video1" interface="aVideo1SoundLevel" role="set"> <bindParam name="var" value="0" /> </bind></pre>

Exemplo 09 – Alternando imagens para identificar ações disponíveis

No exemplo anterior, as imagens dos botões vermelho e verde ficam o tempo todo visíveis, mas somente um dos botões causa efeito ao ser pressionado. Como isto pode causar problemas de usabilidade, o objetivo deste exemplo é modificar a visibilidade das imagens correspondentes aos botões sempre que uma seleção for feita, de modo que apenas o botão que possa causar uma mudança do vídeo seja exibido.

Visões do documento

Visão estrutural

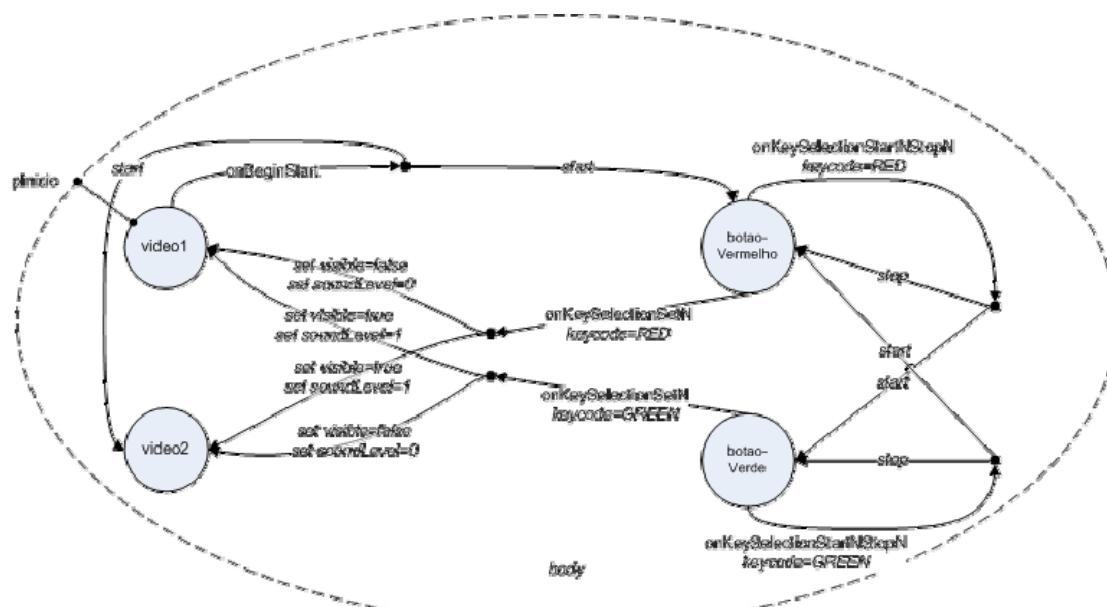


Figura 85. Visão estrutural do exemplo 09.

Observa-se que foram utilizados dois elos para definir o comportamento do documento ao se pressionar cada tecla: *onKeySelectionSetN* e *onKeySelectionStartNStopN*. Uma alternativa seria utilizar um conector que contemplasse todos os papéis, como um conector *onKeySelectionSetNStartNStopN*.

Visão de leiaute

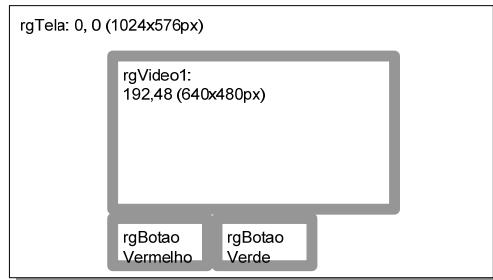


Figura 86. Visão de leiaute do exemplo 09.

Visões temporal e espacial

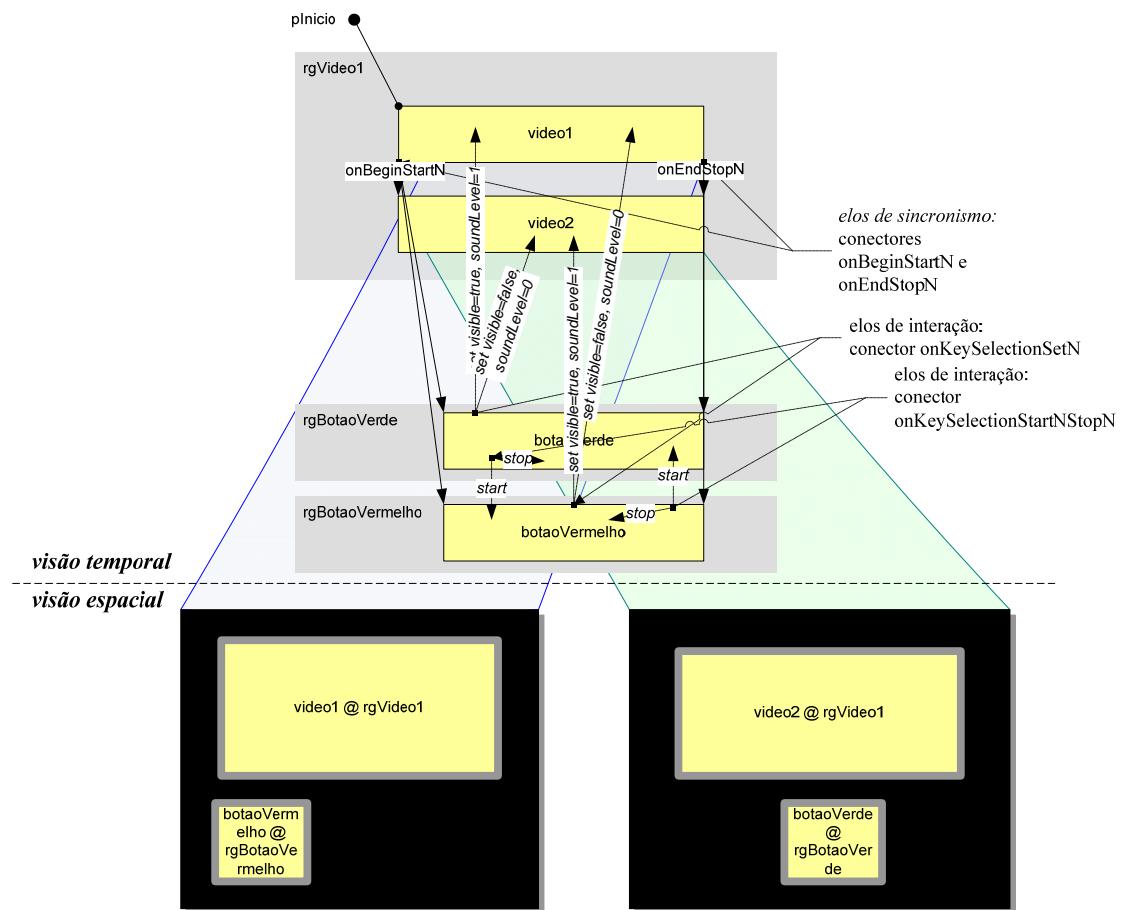


Figura 87. Visões temporal e espacial do exemplo 09.

Em primeiro lugar, o elo de exibição inicial do vídeo e dos botões não deve mais exibir o botão verde, pois pressionar a tecla verde enquanto o vídeo 1 está tocando não produz efeito:

```
<!-- início do video1 deve disparar a exibição do video2 e das imagens do botão vermelho -->

<link id="IVideo_Botoes_start" xconnector="connBase#onBeginStartN">
  <bind component="video1" role="onBegin" />
  <bind component="video2" role="start" />
  <bind component="botaoVermelho" role="start" />
</link>
```

Além disto, a única alteração a ser feita é a inclusão de dois elos, que alternam a exibição das imagens correspondentes aos botões. Esses elos utilizam o conector **onKeySelectionStartNStopN**, especificado para modificar a exibição de um ou mais nós de mídia conforme o pressionamento de uma determinada tecla:

```
<!-- oculta a tecla vermelha quando ela é pressionada -->
<link id="IToggleBotaoVermelho" xconnector="connBase#onKeySelectionStartNStopN">
  <bind component="botaoVermelho" role="onSelection">
    <bindParam name="keyCode" value="RED" />
  </bind>
  <bind component="botaoVerde" role="start" />
  <bind component="botaoVermelho" role="stop" />
</link>

<!-- oculta a tecla verde quando ela é pressionada -->
<link id="IToggleBotaoVerde" xconnector="connBase#onKeySelectionStartNStopN">
  <bind component="botaoVerde" role="onSelection">
    <bindParam name="keyCode" value="GREEN" />
  </bind>
  <bind component="botaoVerde" role="stop" />
  <bind component="botaoVermelho" role="start" />
</link>
```

Adaptação do comportamento do programa

Esta seção ilustra o uso do nó do tipo **settings** (`application/x-ginga-settings`) para armazenar valores que podem ser manipulados pelos elos para modificar o comportamento do programa.

Exemplo 10 – Simulação de um menu de DVD

O objetivo deste exemplo é exibir um vídeo em *loop*, apresentando ao mesmo tempo duas opções de idioma. Ao selecionar uma das opções (pressionando a tecla verde ou vermelha do controle remoto da TV digital), é exibido um vídeo de abertura (independente do idioma selecionado) e, em seguida, um vídeo sem som e um áudio de acordo com o idioma escolhido. Todos os vídeos devem ser apresentados na mesma posição da tela.

Para armazenar a opção selecionada, será utilizado um nó do tipo **settings** com uma propriedade *idioma*. Já para selecionar o nó de áudio a ser reproduzido ao final do *video2*, são definidas regras (**rules**) e um nó **switch**.

Visões do documento

Visão estrutural

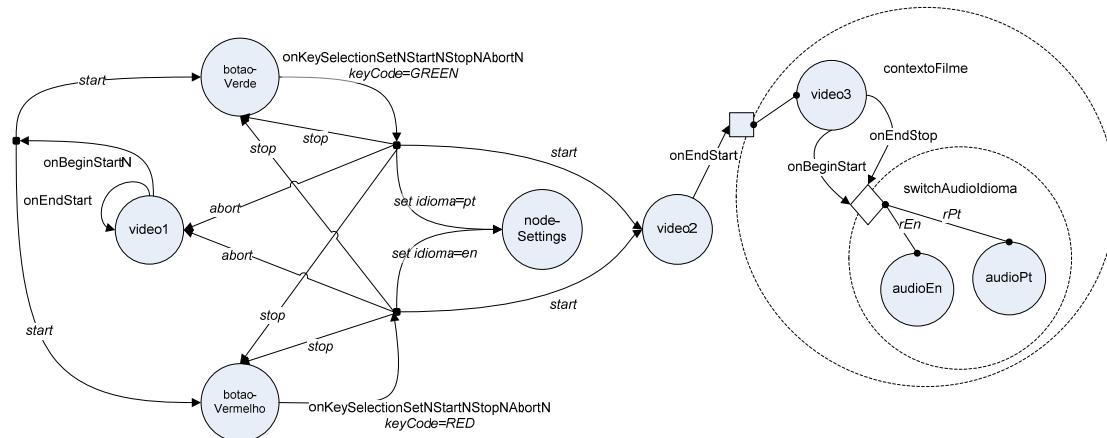


Figura 88. Visão estrutural do exemplo 10.

Quando o botão vermelho ou o verde é selecionado, o idioma correspondente é armazenado na propriedade *idioma*, através do conector **OnKeySelectionSetNStartNStopNAbortN**. Este mesmo conector é o responsável por interromper o *video1* e iniciar a exibição do *video2*. Quando o *video2* termina, o elo *lVideo2StartVideo3*, através do conector **OnEndStart**, ativa o nó de contexto *contextoFilme*. A porta desse contexto mapeia o *video3* que, ao ser iniciado, dispara, através do conector **onBeginStart**, o **switch** *switchAudioIdioma*, que apresentará o áudio conforme o valor do nó **settings**, avaliado pelas regras *rEn* e *rPt*.

Nota-se que deve ser criado um segundo descritor para o *video3* pois, apesar de ele ter as mesmas dimensões dos outros vídeos, ele deve ser apresentado sem som (*soundLevel=0*).

Visão de leiaute

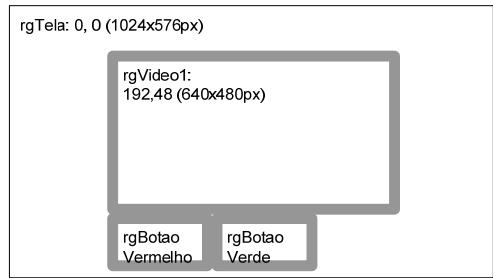


Figura 89. Visão de leiaute do exemplo 10.

Visões temporal e espacial

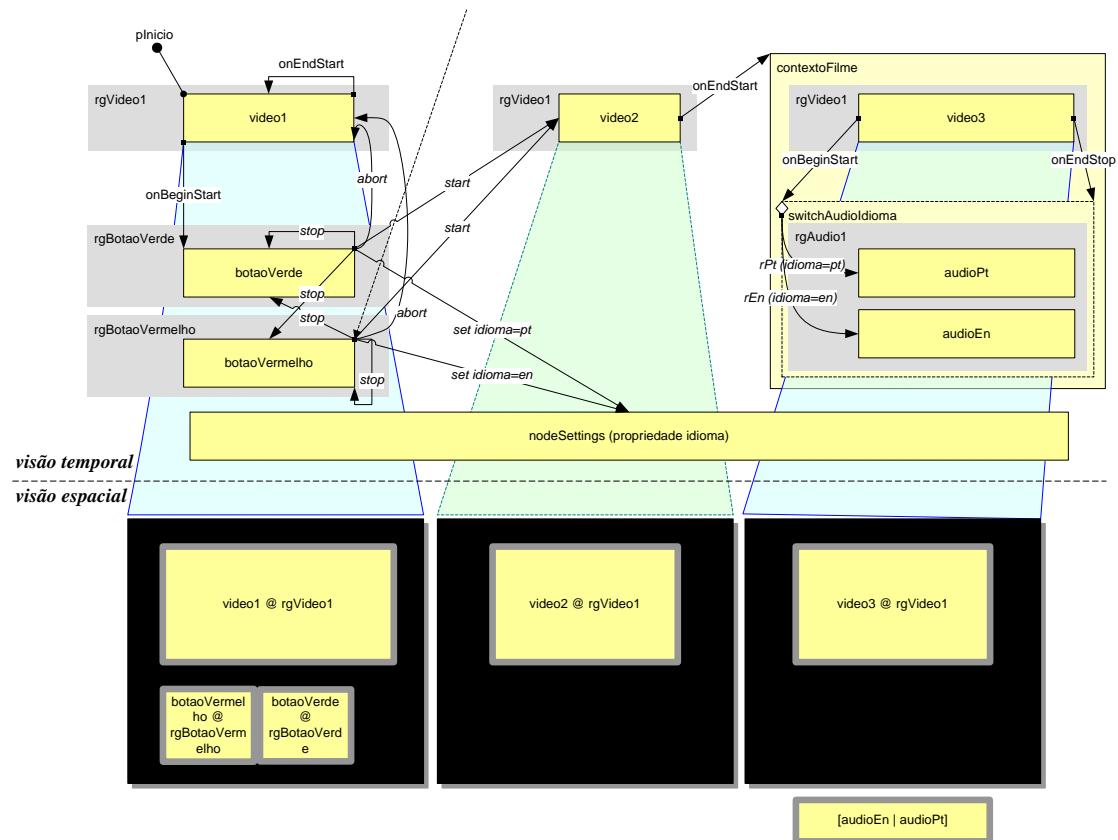


Figura 90. Visões temporal e espacial do exemplo 10.

Para saber mais: 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:

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

Os seguintes operadores podem ser utilizados como **comparator** 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 que ou igual a*);
- **lt** (*less than – menor que*);
- **le** (*less than or equal to – menor que 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>
```

Para saber mais: **Switch**

Um **switch** é um contexto com nós alternativos, ou seja, dentre 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="switchAudioIdioma">
    <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="dAudio1" />
</switch>
```

Nesse exemplo, a mídia *audioEn* será reproduzida caso a regra *rEn* seja válida, ou seja, caso *idioma* possua o valor “en”. Se essa regra não for válida, o próximo mapeamento é avaliado, e assim sucessivamente, até uma regra válida ser alcançada ou até terminar o conjunto de mapeamentos daquele *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>
```

Voltando ao exemplo, o *switchAudioIdioma* é regido pelas regras *rEn* e *rPt*, definidas previamente na seção **ruleBase**. Ao apresentar o switch, o formatador verifica qual é a primeira regra satisfeita e dispara a apresentação do nó correspondente. No exemplo, caso o valor da propriedade *idioma* seja igual a “en”, o áudio apresentado será *audioEn*. Caso a propriedade *idioma* tenha valor igual a “pt”, então o áudio apresentado será *audioPt*.

O conector **onKeySelectionSetNStartNStopNAbortN** é apresentado na Tabela 10:

Tabela 10. Definição do conector onKeySelectionSetNStartNStopNAbortN.

Nome:	onKeySelectionSetNStartNStopNAbortN
Condição:	tecla <keyCode> acionada (papel <i>onSelection</i>)
Ação:	<ul style="list-style-type: none"> ▪ exibe as mídias identificadas pelo papel <i>start</i>; ▪ pára as mídias identificadas pelo papel <i>stop</i>; ▪ aborta as mídias identificadas pelo papel <i>abort</i>; e ▪ define o valor <var> à propriedade mapeada através do papel <i>set</i>.
Código NCL:	<pre><causalConnector id="onKeySelectionSetNStartNStopNAbortN"> <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></pre>
Observação:	<p>Os elos que utilizarem este conector deverão identificar, como parâmetro adicional do elo (linkParam) ou como parâmetro do mapeamento de um nó (bindParam), o código virtual da tecla do controle remoto associada à seleção.</p> <p>Por exemplo:</p> <pre><linkParam name="keyCode" value="VK_ENTER" /></pre> <p>Os elos que utilizarem este conector deverão identificar também, como parâmetro adicional dos mapeamentos dos nós (bindParam), os novos valores das propriedades. Por exemplo:</p> <pre><bindParam name="var" value="en" /></pre>

Exemplo 11 – Simulação de um menu de DVD com feedback

No exemplo anterior, ao selecionar um idioma, o usuário não recebe qualquer *feedback* imediato sobre qual opção foi selecionada. Ele precisa esperar o início da reprodução do áudio, após o *video2*, para saber se fez a seleção correta. Como boa prática de projeto de programa audiovisual interativo, deve-se fornecer um *feedback* para toda ação do usuário. Uma forma de se fazer isso é ocultar momentaneamente as opções que não foram selecionadas, deixando apenas a seleção do usuário visível, mesmo que por uma fração de segundo. Para atingir esse efeito, pode-se definir um atributo *delay* no mapeamento de ativação dos papéis do conector.

Visões do documento

Visão estrutural

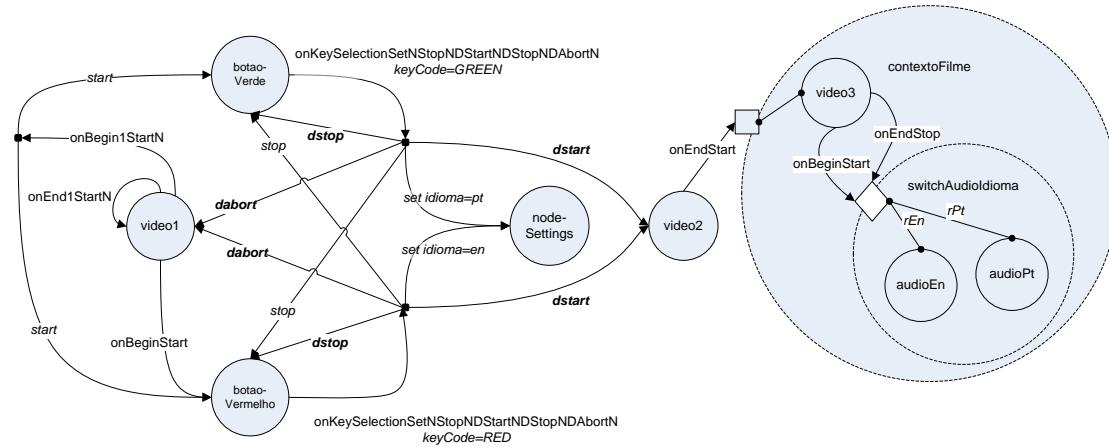


Figura 91. Visão estrutural do exemplo 11.

Visões temporal e espacial

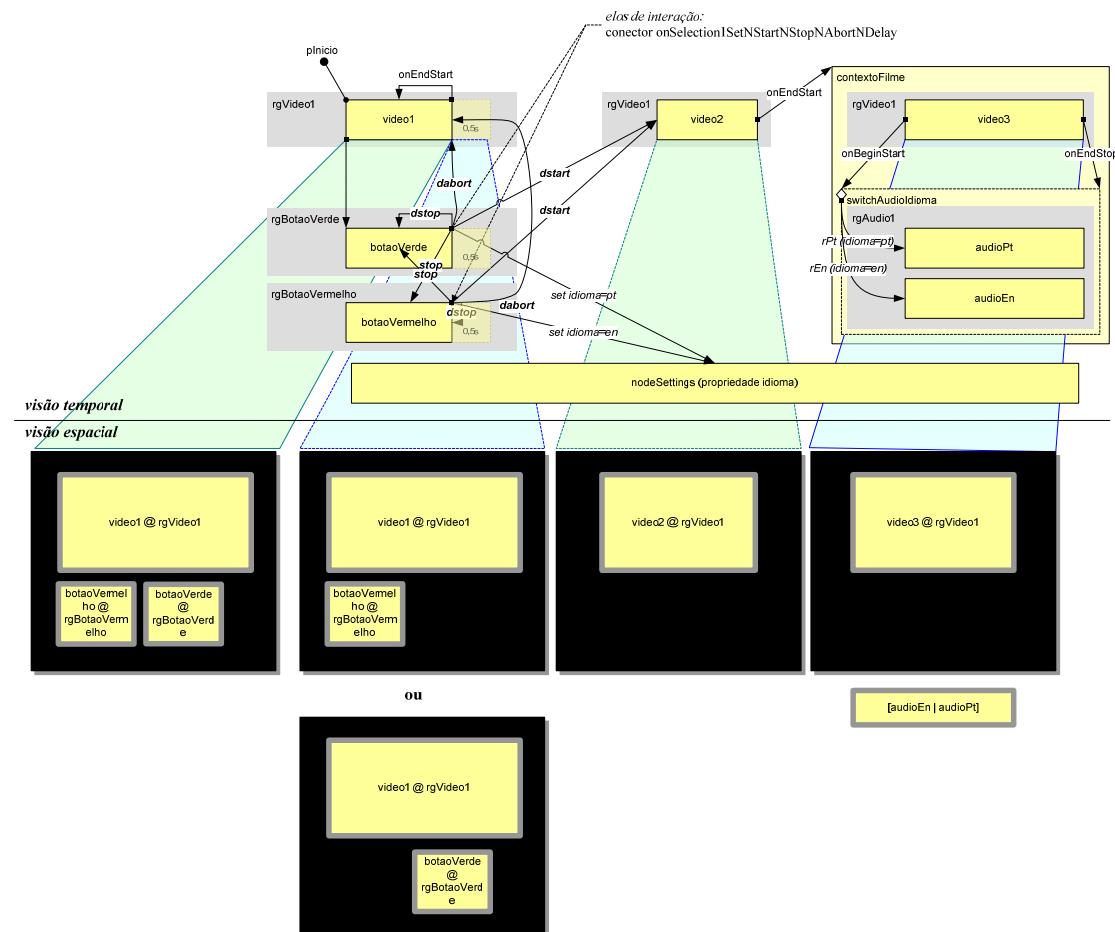


Figura 92. Visões temporal e espacial do exemplo 11.

A Tabela 11 apresenta o conector **onKeySelectionSetNStopNDStartNDStopNDAbortN**, criado para acomodar não apenas o disparo imediato de eventos, mas também um disparo retardado por meio segundo:

Tabela 11. Redefinição do conector onKeySelectionSetNStopNDStartNDStopNDAbortN, para introduzir papéis de ativação com retardo.

Nome:	onKeySelectionSetNStopNDStartNDStopNDAbortN
Condição:	tecla <keyCode> acionada (papel <i>onSelection</i>)
Ação:	<ul style="list-style-type: none"> ▪ pára as mídias identificadas pelo papel <i>stop</i>; ▪ define o valor <var> à propriedade mapeada através do papel <i>set</i>; ▪ exibe, após um retardo de 0,5s, as mídias identificadas pelo papel <i>dstart</i>; ▪ pára, após um retardo de 0,5s, as mídias identificadas pelo papel <i>dstop</i>; e ▪ aborta, após um retardo de 0,5s, as mídias identificadas pelo papel <i>dabort</i>.
Código NCL:	<pre><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></pre>

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. Por exemplo:

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

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

Menu

Exemplo 12 – Seleção através das setas do controle remoto

Os exemplos de interatividade apresentados até agora fizeram uso dos botões coloridos. No entanto, essa estratégia está limitada ao número de botões coloridos do controle remoto da TV, que geralmente são quatro. Como existem situações em que são oferecidas aos usuários mais do que quatro opções, torna-se necessário apresentar outra estratégia de seleção. Uma estratégia utilizada com freqüência faz uso das setas do controle remoto para identificar uma opção, seguidas do pressionamento da tecla “OK” para ativar o elo correspondente à seleção realizada.

Para realizar esse tipo de menu, numa organização vertical, segue-se o esquema da Figura 93. O esquema considera que as opções estão ordenadas. Inicia-se com a primeira opção em foco. Caso a tecla para baixo seja pressionada, a próxima opção recebe o foco. Já a tecla para cima é utilizada para passar o foco para a opção anterior na lista. Observa-se que o esquema não representa um menu circular, ou seja, pressionar a seta para baixo na última opção não seleciona a primeira, assim como pressionar a seta para cima na primeira opção não seleciona a última. Após selecionar a opção desejada, o usuário pode pressionar a tecla “OK” (código virtual VK_ENTER) para acionar o elo correspondente.

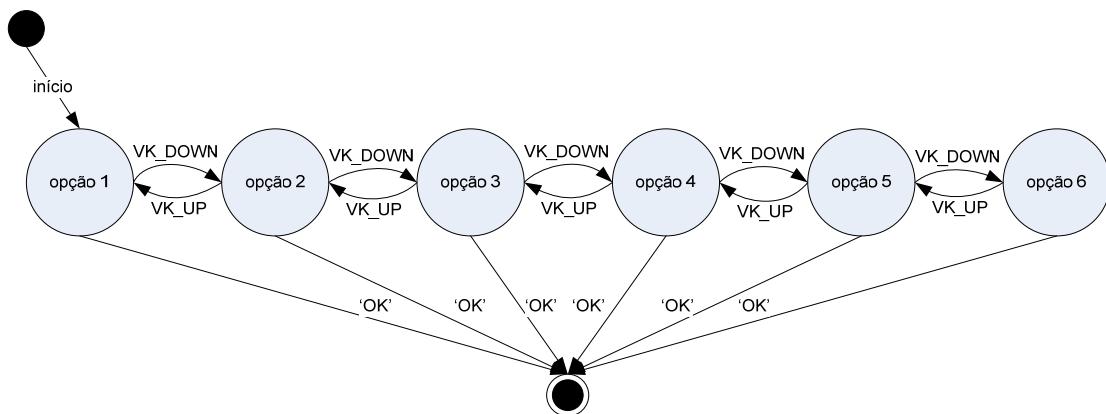


Figura 93. Esquema de seleção de uma dentre 6 opções de menu.

A navegação por teclas é definida nos descritores dos diversos botões. A cada botão deve ser atribuído um *focusIndex*, que nesse caso varia de 1 a 6. Os atributos *moveDown* e *moveUp* de cada descritor indicam para qual opção deve mudar o foco quando as teclas DOWN e UP forem pressionadas, respectivamente.

Nesse tipo de menu, é imprescindível que o usuário seja mantido informado sobre qual é a seleção atual. Para isso, este exemplo utiliza o conceito de foco introduzido em NCL 3.0. Pode-se definir uma cor e largura de borda do elemento em foco pelos atributos *focusBorderColor* e *focusBorderWidth*. Um valor negativo (e.g. -2) para a largura indica que

a borda deve ser exibida ocupando alguns (no caso, 2) pixels dentro da região em que o botão aparece, enquanto um valor positivo projeta a borda para fora dessa região. Neste exemplo, uma borda retangular aparece ao redor da imagem que está em foco, conforme o esquema da Figura 94.

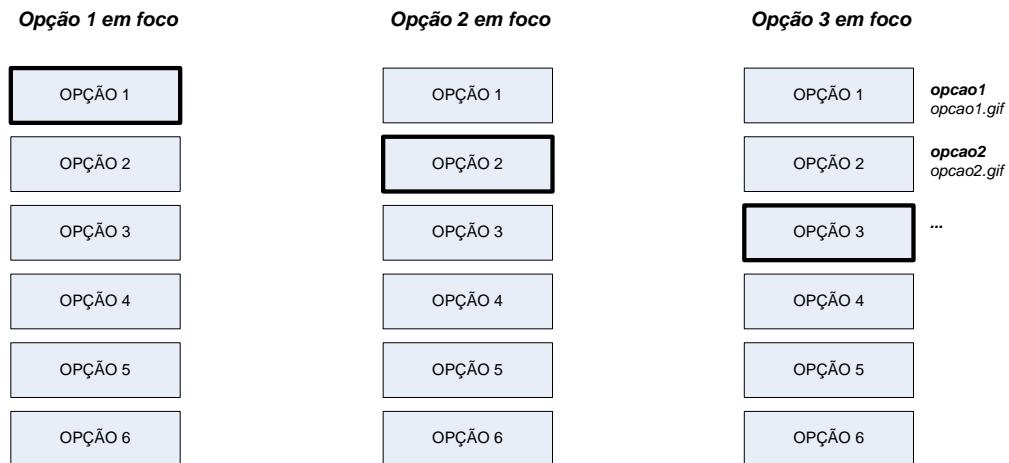


Figura 94. Indicação da opção atual selecionada.

Esse exemplo utiliza novamente uma propriedade do nó do tipo **settings** (*nodeSettings*), desta vez chamado *opcao*. O valor dessa propriedade é utilizada pelas regras que definem qual nó deve ser selecionado pelo do nó de alternativa *switchOpcão*.

Este exemplo faz uso do conector predefinido **onBeginStart** e do conector **onSelectionSetNStartNStopN** descrito na Tabela 12 a seguir.

Tabela 12. Definição do conector *onSelectionSetNStartNStopN*, para parar os botões e exibir o nó correspondente à opção selecionada.

Nome: **onSelectionSetNStartNStopN**

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*; e
- pára as mídias identificadas pelo papel *stop*.

Código

```
<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>
```

NCL:

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). Por exemplo:

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

Re-uso de nós

Existem casos em que um mesmo nó é utilizado em diferentes partes de um documento. No exemplo a seguir, temos um nó de vídeo que pode ser apresentado em dois “modos” diferentes: um modo avançado, com alguns textos sincronizados com o vídeo, ou um modo básico, no qual, além de textos sincronizados, alguns vídeos de apoio interrompem o vídeo principal para esclarecer melhor algum ponto, para depois retomar o vídeo principal.

Para se definir um programa assim, pode-se definir um nó com o vídeo principal e todas suas características (arquivo de mídia associado, descritor e outras propriedades), e apenas se referir a ele em outros contextos, conforme ilustrado pela Figura 95.

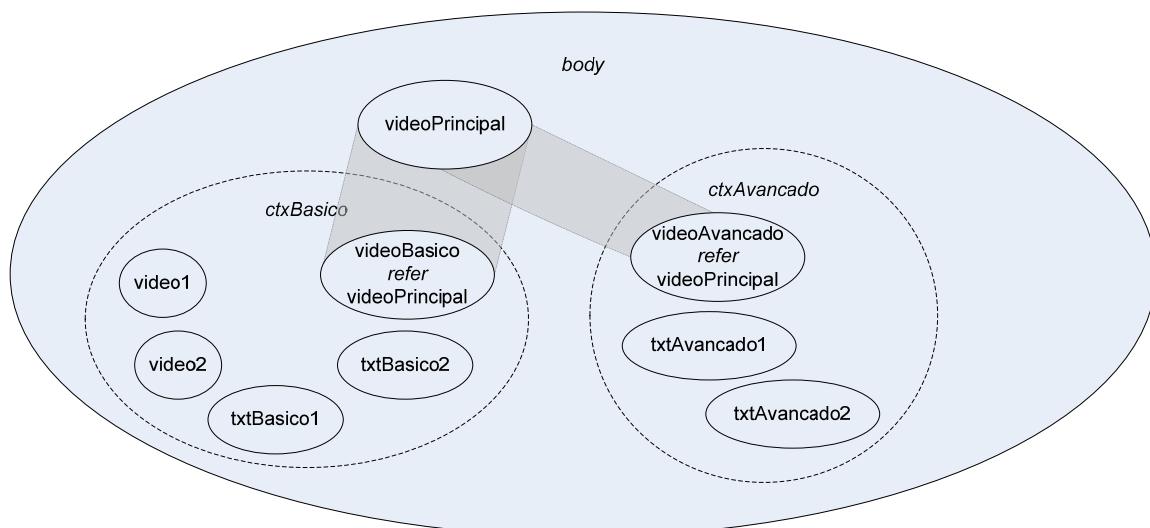


Figura 95. Visão estrutural parcial de um nó videoPrincipal sendo reutilizado em dois outros contextos.

Para se reutilizar um nó, deve-se utilizar o atributo **refer** na definição do nó que faz a referência. No exemplo da figura acima, teríamos

```

<body>
    <media id= "videoPrincipal" src= "princ.mpg" descriptor= "dPrincipal" />
    <context id= "ctxBasico">
        <media id= "videoBasico" refer= "videoPrincipal" />
        ...
    </context>
    <context id= "ctxAvancado">
        <media id= "videoAvancado" refer= "videoPrincipal" />
        ...
    </context>
    ...
</body>

```

É importante observar que, apesar de terem identificadores diferentes, eles são o “mesmo” nó. Como existe uma restrição no modelo NCM subjacente à NCL que impede que o mesmo nó apareça mais do que uma vez num mesmo contexto, não se pode fazer referência para um nó

no mesmo contexto, nem ter duas referências, num único contexto, apontando para um nó em outro, conforme ilustrado pela Figura 96.

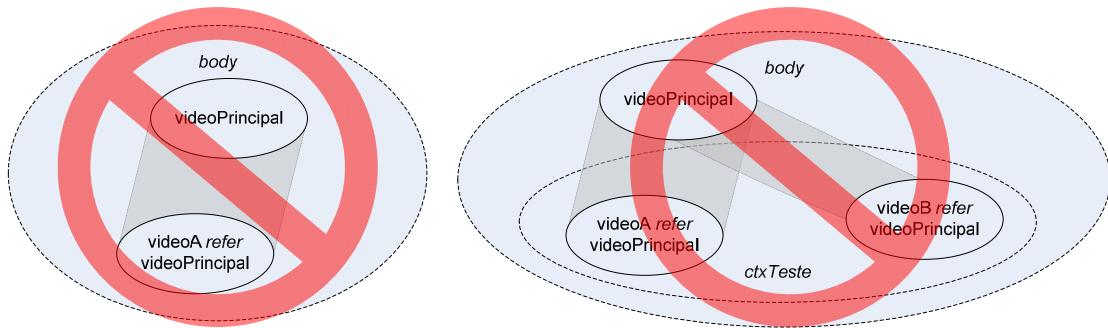


Figura 96. Situações inválidas de re-uso de nó.

A Listagem 13 apresenta o código de um exemplo de reutilização de nós.

Apêndice I – Listagens dos exemplos

Exemplo 01 – Seu primeiro documento NCL: reprodução de um objeto de mídia

Para construir um programa em NCL que reproduz um vídeo, é necessário criar os seguintes elementos (indicados na Listagem 1):

1. ❶ a região da tela que define o dispositivo onde o programa será exibido (região *rgTV*, linhas 41-43)
2. ❷ a região da tela onde o vídeo será exibido (região *rgVideo1*, linha 42);
3. ❸ o descritor que determina a forma como o vídeo será exibido (descritor *dVideo1*, linha 52). Nesse caso, o vídeo associado a este descritor deve ser exibido na região *rgVideo1*, com as propriedades *default* de exibição de vídeo;
4. ❹ o nó de mídia, o vídeo propriamente dito (mídia *video1*, linha 76); e
5. ❺ a porta que define o ponto de entrada do documento hipermídia. Nesse caso, a porta *pInicio* (linha 69), que mapeia para o primeiro elemento de mídia, *video1*.

Observação: Para executar este exemplo, é necessário que haja um vídeo chamado *video1.mpg* no subdiretório *media* sob o diretório onde o arquivo NCL estiver localizado, como a seguir:

```
exemplo01.ncl
media/
    video1.mpg
```

Listagem 1. Documento NCL para reprodução de um só vídeo.

```
1:  <?xml version="1.0" encoding="ISO-8859-1"?>
2:
3:  <!-----+
4: ! EXEMPLO 01
5: !
6: ! Objetivo: Reproduzir um vídeo no meio da tela.
7: !
8: ! Características:
9: !
10: ! - sincronismo: nenhum
11: ! - interação do usuário: nenhuma
12: !
13: ! Preparação:
14: !
15: ! Para executar este exemplo, é necessário ter a seguinte mídia no subdiretório
16: ! mídia a partir do caminho do arquivo NCL:
17: !
18: ! 1) arquivo de vídeo chamado video1.mpg
19: !
20: !+-----+
21:
22: <!-----+
```

```

23: ! CABEÇALHO NCL:
24: ! define as URIs dos esquemas da NCL
25: !+++++>
26:
27: <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">
28:
29: <!--+++++>
30: ! CABEÇALHO DO PROGRAMA
31: !+++++>
32:
33: <head>
34:
35: <!--+++++>
36: ! BASE DE REGIÕES:
37: ! define as regiões na tela onde as mídias são apresentadas
38: !+++++>
39:
40: <regionBase>
41:   <region id="rgTV" height="576" width="1024">
42:     <region id="rgVideo1" height="480" width="640" top="48" left="192"/>
43:   </region>
44: </regionBase>
45:
46: <!--+++++>
47: ! BASE DE DESCRIPTORES:
48: ! define como as mídias são apresentadas
49: !+++++>
50:
51: <descriptorBase>
52:   <descriptor id="dVideo1" region="rgVideo1" />
53: </descriptorBase>
54:
55: </head>
56:
57: <!--+++++>
58: ! CORPO DO PROGRAMA:
59: ! define as mídias e estrutura do programa
60: !+++++>
61:
62: <body>
63:
64: <!--+++++>
65: ! PONTO DE ENTRADA:
66: ! indica o componente onde o programa inicia
67: !+++++>
68:
69: <port id="plinicio" component="video1"/> 4
70:
71: <!--+++++>
72: ! MÍDIAS:
73: ! define o local dos arquivos de mídia e as associa com seus descritores
74: !+++++>
75:
76: <media type="video/mpeg" id="video1" src="media/video1.mpg" descriptor="dVideo1"/> 5
77:
78: </body>
79: </ncl>

```

Exemplo 02 – Iniciando e terminando dois objetos de mídia simultaneamente

Observação: Para executar este exemplo, é necessário que haja, no subdiretório *media*, sob o diretório onde o arquivo NCL estiver localizado, as seguintes mídias:

- um arquivo TXT chamado *titulo1.txt*; e
- um vídeo chamado *video1.mpg*.

Assim, a estrutura de diretórios para este exemplo deve ser a seguinte:

```
exemplo02.ncl
maestroConnectorBase.conn (arquivo de base de conectores predefinidos que vem com o Composer)
media/
    video1.mpg
    titulo1.txt
```

A Listagem 2 apresenta o código do exemplo. Esse código toma como base o exemplo anterior, e as linhas em negrito indicam aquelas que foram inseridas ou modificadas para este exemplo.

Como no exemplo anterior, são inseridos:

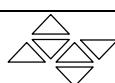
1. **①** a região onde será exibido o título (*rgTitulo1*, linha 44);
2. **②** o descritor que define como e onde o nó de mídia será exibido (*dTitulo1*, linha 55); e
3. **④** o nó de mídia propriamente dito, apontando para o arquivo TXT do conteúdo (*titulo1*, linha 92).

Além disto, foram introduzidos dois novos trechos ao programa:

4. **③** importação da base de conectores atualmente definidos, a partir do arquivo *maestroConnectorBase.conn* (seção **connectorBase**, linhas 63 a 65); e
5. **⑤** dois elos de ligação entre o nó *video1* e o nó *titulo1*, fazendo uso dos conectores **onBegin1StartN** e **onEnd1StopN**:
 - o elo *lBeginVideo1StartTitulo1* (conector **onBeginStart**) define o início do nó *titulo1* a partir do início do nó *video1* (linhas 98 a 101); e
 - o elo *lEndVideo1StopTitulo1* (conector **onEndStop**) define o término do nó *titulo1* a partir do término do nó *video1* (linhas 105 a 108).

Listagem 2. Documento NCL com elos para sincronizar o início e o término de exibição de mídias.

```
1:  <?xml version="1.0" encoding="ISO-8859-1"?>
2:
3:  <!-----+
4:  EXEMPLO 02
5:
6:  Objetivo: Reproduzir um título e um vídeo numa região da tela, sincronizados.
7:
```



```

8:    Características:
9:
10:   - sincronismo: simples (de início e término de uma mídia)
11:   - interação do usuário: nenhuma
12:
13:   Preparação:
14:
15:   Para executar este exemplo, é necessário ter as seguintes mídias no subdiretório
16:   mídia a partir do caminho do arquivo NCL:
17:
18:   1) arquivo TXT com o título, chamado titulo1.txt
19:   2) arquivo de vídeo chamado video1.mpg
20:
21:   !+++++>
22:
23:   <!--+
24:   ! CABEÇALHO NCL:
25:   ! define as URIs dos esquemas da NCL
26:   !+++++>
27:
28:   <ncl id="exemplo02" 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">
29:
30:   <!--+
31:   ! CABEÇALHO DO PROGRAMA
32:   !+++++>
33:
34:   <head>
35:
36:   <!--+
37:   ! BASE DE REGIÕES:
38:   ! define as regiões na tela onde as mídias são apresentadas
39:   !+++++>
40:
41:   <regionBase>
42:     <region id="rgTV" height="576" width="1024">
43:       <region id="rgVideo1" height="480" width="640" top="48" left="192"/>
44:       <region id="rgTitulo1" height="28" width="640" top="20" left="192"/>
45:     </region>
46:   </regionBase>
47:
48:   <!--+
49:   ! BASE DE DESCRIPTORES:
50:   ! define como as mídias são apresentadas
51:   !+++++>
52:
53:   <descriptorBase>
54:     <descriptor id="dVideo1" region="rgVideo1" />
55:     <descriptor id="dTítulo1" region="rgTitulo1" />
56:   </descriptorBase>
57:
58:   <!--+
59:   ! BASE DE CONECTORES:
60:   ! definem o comportamento dos elos
61:   !+++++>
62:
63:   <connectorBase>
64:     <importBase alias="connBase" documentURI="maestroConnectorBase.conn" />
65:   </connectorBase>

```

1

2

3

```
66:  
67: </head>  
68:  
69: <!--+-----+-----+-----+-----+-----+-----+-----+  
70: ! CORPO DO PROGRAMA:  
71: ! define as mídias e estrutura do programa  
72: !+-----+-----+-----+-----+-----+-----+----->  
73:  
74: <body>  
75:  
76: <!--+-----+-----+-----+-----+-----+-----+-----+  
77: ! PONTO DE ENTRADA:  
78: ! indica o componente onde o programa inicia  
79: !+-----+-----+-----+-----+-----+-----+----->  
80:  
81: <port id="pInicio" component="video1" />  
82:  
83: <!--+-----+-----+-----+-----+-----+-----+-----+  
84: ! MÍDIAS:  
85: ! define o local dos arquivos de mídia e as associa com seus descritores  
86: !+-----+-----+-----+-----+-----+-----+----->  
87:  
88: <media type="video/mpeg" id="video1" src="media/video1.mpg" descriptor="dVideo1" />  
89: <media type="text/plain" id="titulo1" src="media/titulo1.txt" descriptor="dTítulo1" />  
90:  
91: <!--+-----+-----+-----+-----+-----+-----+-----+  
92: ! ELOS  
93: ! define os elos que regem o sincronismo entre as mídias  
94: !+-----+-----+-----+-----+-----+-----+----->  
95:  
96: <!-- titulo1 deve ser iniciado simultaneamente a video1 -->  
97:  
98: <link id="IBeginVideo1StartTítulo1" xconnector="connBase#onBeginStart">  
99:   <bind component="video1" role="onBegin" />  
100:  <bind component="título1" role="start" />  
101: </link>  
102:  
103: <!-- títuo1 deve ser terminado simultaneamente a video1 -->  
104:  
105: <link id="IEndVideo1StopTítulo1" xconnector="connBase#onEndStop">  
106:   <bind component="video1" role="onEnd" />  
107:   <bind component="título1" role="stop" />  
108: </link>  
109:  
110: </body>  
111: </ncl>
```

4

5

Exemplo 03 – Iniciando um objeto de mídia quando outro termina

Observação: Para executar este exemplo, é necessário que haja, no subdiretório *media*, sob o diretório onde os arquivos NCL (exemplo03.ncl e maestroConnectorBase.conn) estiverem localizados, dois arquivos de vídeo, chamados *video1.mpg* e *video2.mpg*, como indicado a seguir:

```
exemplo03.ncl
maestroConnectorBase.conn
media\
    video1.mpg
    video2.mpg
```

A Listagem 3 apresenta o código do exemplo. Esse código toma como base o exemplo 01, e as linhas em negrito indicam aquelas que foram inseridas ou modificadas para este exemplo.

Como nos exemplos anteriores, **①** o nó de mídia é inserido, apontando para o segundo arquivo de vídeo (*video2*, linha 86). Além disto, **②** foi introduzido um novo elo de ligação, *lEndVideo1StartVideo2*, que dispara o nó de vídeo *video2* assim que o nó de vídeo *video1* tem sua exibição terminada (linhas 95 a 98).

Observa-se que, como o nó de vídeo deveria ser apresentado da mesma forma e na mesma região, não foi necessário criar um novo descritor nem uma nova região para o novo nó. Isto é uma evidência de que o custo de certos tipos de extensão a um programa existente pode ser extremamente baixo.

Listagem 3. Documento NCL com elos para sincronizar o início de exibição de uma mídia quando outra termina.

```
1:  <?xml version="1.0" encoding="ISO-8859-1"?>
2:
3:  <!-----+
4:  EXEMPLO 03
5:
6:  Objetivo: Reproduzir dois vídeos em seqüência, numa mesma região da tela.
7:
8:  Características:
9:
10: - sincronismo: término de uma mídia -> início de outra
11: - interação do usuário: nenhuma
12:
13: Preparação:
14:
15: Para executar este exemplo, é necessário ter as seguintes mídias no subdiretório
16: mídia a partir do caminho do arquivo NCL:
17:
18: 1) arquivo de vídeo chamado video1.mpg
19: 2) arquivo de vídeo chamado video2.mpg
20: !-----+
21:
22: <!-----+
```

```

23: ! CABEÇALHO NCL:
24: ! define as URIs dos esquemas da NCL
25: !+++++>
26:
27: <ncl id="exemplo03" 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">
28:
29: <!--+++++>
30: ! CABEÇALHO DO PROGRAMA
31: !+++++>
32:
33: <head>
34:
35: <!--+++++>
36: ! BASE DE REGIÕES:
37: ! define as regiões na tela onde as mídias são apresentadas
38: !+++++>
39:
40: <regionBase>
41:   <region id="rgTV" height="576" width="1024">
42:     <region id="rgVideo1" height="480" width="640" top="48" left="192"/>
43:   </region>
44: </regionBase>
45:
46: <!--+++++>
47: ! BASE DE DESCRIPTORES:
48: ! define como as mídias são apresentadas
49: !+++++>
50:
51: <descriptorBase>
52:   <descriptor id="dVideo1" region="rgVideo1" />
53: </descriptorBase>
54:
55: <!--+++++>
56: ! BASE DE CONECTORES:
57: ! definem o comportamento dos elos
58: !+++++>
59:
60: <connectorBase>
61:   <importBase alias="connBase" documentURI="maestroConnectorBase.conn"/>
62: </connectorBase>
63:
64: </head>
65:
66: <!--+++++>
67: ! CORPO DO PROGRAMA:
68: ! define as mídias e estrutura do programa
69: !+++++>
70:
71: <body>
72:
73: <!--+++++>
74: ! PONTO DE ENTRADA:
75: ! indica o componente onde o programa inicia
76: !+++++>
77:
78: <port id="plinicio" component="video1" />
79:
80: <!--+++++>

```

```
81: ! MÍDIAS:  
82: ! define o local dos arquivos de mídia e as associa com seus descritores  
83: !+++++  
84:  
85: <media type="video/mpeg" id="video1" src="media/video1.mpg" descriptor="dVideo1" />  
86: <media type="video/mpeg" id="video2" src="media/video2.mpg" descriptor="dVideo1" />  
87:  
88: <!--+++++  
89: ! ELOS  
90: ! define os elos que regem o sincronismo entre as mídias  
91: !+++++  
92:  
93: <!-- video2 deve ser iniciado quando video1 terminar -->  
94:  
95: <link id="IEndVideo1StartVideo2" xconnector="connBase#onEndStart">  
96:   <bind component="video1" role="onEnd" />  
97:   <bind component="video2" role="start" />  
98: </link>  
99:  
100: </body>  
101: </ncl>
```

1

2

Exemplo 04 – Sincronizando um vídeo com diferentes arquivos de legenda

Observação: Para executar este exemplo, é necessário que haja, no subdiretório *media* sob o diretório onde o arquivo NCL estiver localizado, as seguintes mídias:

- um vídeo chamado *video1.mpg*;
- três arquivos HTML chamados *legenda01.html*, *legenda02.html* e *legenda03.html*, cada qual com um texto a ser sincronizado.

A listagem a seguir apresenta o código necessário para reproduzir o vídeo com a legenda sincronizada. A ① definição das âncoras (linhas 91 a 93) e ② seu uso nos elos (linhas 108, 115, 122, 129, 136 e 143) aparecem em negrito.

Listagem 4. Documento NCL para reprodução de um vídeo com três trechos de legenda sincronizados.

```
1:  <?xml version="1.0" encoding="ISO-8859-1"?>
2:
3:  <!-----+
4:  EXEMPLO 04
5:
6:  Objetivo: Reproduzir um vídeo numa região da tela, sincronizando com trechos
7:  de legenda.
8:
9:  Características:
10:
11: - sincronismo: simples (de início e término de mídias)
12: - interação do usuário: nenhuma
13:
14: Preparação:
15:
16: Para executar este exemplo, é necessário ter as seguintes mídias no subdiretório
17: mídia a partir do caminho do arquivo NCL:
18:
19: 1) arquivo de vídeo chamado video1.mpg
20: 2) 3 arquivos HTML com a legenda, chamados legenda01.html, legenda02.html e
21:    legenda03.html
22: !-----+
23:
24: <!-----+
25: ! CABEÇALHO NCL:
26: ! define as URLs dos esquemas da NCL
27: !-----+
28:
29: <ncl id="exemplo04" 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">
30:
31: <!-----+
32: ! CABEÇALHO DO PROGRAMA
33: !-----+
34:
35: <head>
```

```

36:
37: <!--+-----+
38: ! BASE DE REGIÕES:
39: ! define as regiões na tela onde as mídias são apresentadas
40: +----->
41:
42: <regionBase>
43:   <region id="rgTV" height="576" width="1024">
44:     <region id="rgVideo1" height="480" width="640" top="48" left="192"/>
45:     <region id="rgLegenda" height="28" width="640" top="528" left="192"/>
46:   </region>
47: </regionBase>
48:
49: <!--+-----+
50: ! BASE DE DESCRIPTORES:
51: ! define como as mídias são apresentadas
52: +----->
53:
54: <descriptorBase>
55:   <descriptor id="dVideo1" region="rgVideo1" />
56:   <descriptor id="dLegenda" region="rgLegenda" />
57: </descriptorBase>
58:
59: <!--+-----+
60: ! BASE DE CONECTORES:
61: ! definem o comportamento dos elos
62: +----->
63:
64: <connectorBase>
65:   <importBase alias="connBase" documentURI="maestroConnectorBase.conn"/>
66: </connectorBase>
67:
68: </head>
69:
70: <!--+-----+
71: ! CORPO DO PROGRAMA:
72: ! define as mídias e estrutura do programa
73: +----->
74:
75: <body>
76:
77: <!--+-----+
78: ! PONTO DE ENTRADA:
79: ! indica o componente onde o programa inicia
80: +----->
81:
82: <port id="plnicio" component="video1" />
83:
84: <!--+-----+
85: ! MÍDIAS:
86: ! define o local dos arquivos de mídia e as associa com seus descritores
87: +----->
88:
89: <media type="video/mpeg" id="video1" src="media/video1.mpg" descriptor="dVideo1">
90:   <!-- âncoras no vídeo que devem ser sincronizadas com a legenda -->
91:   <area id="aVideoLegenda01" begin="5s" end="9s"/>
92:   <area id="aVideoLegenda02" begin="10s" end="14s"/>
93:   <area id="aVideoLegenda03" begin="15s" end="19s"/>
94: </media>
95:
```

```

96: <media type="text/html" id="legenda01" src="media/legenda01.html" descriptor="dLegenda" />
97: <media type="text/html" id="legenda02" src="media/legenda02.html" descriptor="dLegenda" />
98: <media type="text/html" id="legenda03" src="media/legenda03.html" descriptor="dLegenda" />
99:
100: <!--+++++-----+
101: ! ELOS
102: ! define os elos que regem o sincronismo entre as mídias
103: !-----+
104:
105: <!-- início da legenda 01 -->
106:
107: <link id="lLegenda01_start" xconnector="connBase#onBeginStart">
108:   <bind component="video1" interface="aVideoLegenda01" role="onBegin" />
109:   <bind component="legenda01" role="start" />
110: </link>
111:
112: <!-- término da legenda 01 -->
113:
114: <link id="lLegenda01_stop" xconnector="connBase#onEndStop">
115:   <bind component="video1" interface="aVideoLegenda01" role="onEnd" />
116:   <bind component="legenda01" role="stop" />
117: </link>
118:
119: <!-- início da legenda 02 -->
120:
121: <link id="lLegenda02_start" xconnector="connBase#onBeginStart">
122:   <bind component="video1" interface="aVideoLegenda02" role="onBegin" />
123:   <bind component="legenda02" role="start" />
124: </link>
125:
126: <!-- término da legenda 02 -->
127:
128: <link id="lLegenda02_stop" xconnector="connBase#onEndStop">
129:   <bind component="video1" interface="aVideoLegenda02" role="onEnd" />
130:   <bind component="legenda02" role="stop" />
131: </link>
132:
133: <!-- início da legenda 03 -->
134:
135: <link id="lLegenda03_start" xconnector="connBase#onBeginStart">
136:   <bind component="video1" interface="aVideoLegenda03" role="onBegin" />
137:   <bind component="legenda03" role="start" />
138: </link>
139:
140: <!-- término da legenda 03 -->
141:
142: <link id="lLegenda03_stop" xconnector="connBase#onEndStop">
143:   <bind component="video1" interface="aVideoLegenda03" role="onEnd" />
144:   <bind component="legenda03" role="stop" />
145: </link>
146:
147: </body>
148: </ncl>

```

Exemplo 05 – Sincronizando um vídeo com um único arquivo de legenda, segmentado

Observação: Para executar este exemplo, é necessário que haja, no subdiretório *media* sob o diretório onde o arquivo NCL estiver localizado, as seguintes mídias:

- um vídeo chamado *video1.mpg*; e
- um arquivo HTML chamados *legendas.html*, contendo três elementos DIV identificados por *legenda01*, *legenda02* e *legenda03*, cada qual com um texto a ser sincronizado, como por exemplo:

```
<html>
<body>
    <div id="legenda01">Legenda 1</div>
    <div id="legenda02">Legenda 2</div>
    <div id="legenda03">Legenda 3</div>
</body>
</html>
```

A listagem a seguir apresenta o código necessário para reproduzir o vídeo com a legenda sincronizada. O que muda aqui, com relação ao exemplo anterior, é **①** a forma de definição das mídias de legenda.

Listagem 5. Documento NCL para reprodução de um vídeo com três trechos de legenda sincronizados, num único arquivo.

```
149: <?xml version="1.0" encoding="ISO-8859-1"?>
150:
151: <!-----+
152: EXEMPLO 05
153:
154: Objetivo: Reproduzir um vídeo numa região da tela, sincronizando com trechos
155: de legenda.
156:
157: Características:
158:
159: - sincronismo: simples (de início e término de mídias)
160: - interação do usuário: nenhuma
161:
162: Preparação:
163:
164: Para executar este exemplo, é necessário ter as seguintes mídias no subdiretório
165: mídia a partir do caminho do arquivo NCL:
166:
167: 1) arquivo de vídeo chamado video1.mpg
168: 2) um único arquivo HTML chamado legendas.html, com três legendas, cada qual num
169:     elemento DIV (com IDs legenda01, legenda02 e legenda03, respectivamente)
170:
171: !-----+
172:
173: <!-----+
174: ! CABEÇALHO NCL:
175: ! define as URIs dos esquemas da NCL
176: !-----+
177:
178: <ncl id="exemplo05" 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">

179:
180: <!--+-----+
181: ! CABEÇALHO DO PROGRAMA
182: !-----+
183:
184: <head>
185:
186: <!--+-----+
187: ! BASE DE REGIÕES:
188: ! define as regiões na tela onde as mídias são apresentadas
189: !-----+
190:
191: <regionBase>
192:   <region id="rgTV" height="576" width="1024">
193:     <region id="rgVideo1" height="480" width="640" top="48" left="192"/>
194:     <region id="rgLegenda" height="28" width="640" top="528" left="192"/>
195:   </region>
196: </regionBase>
197:
198: <!--+-----+
199: ! BASE DE DESCRIPTORES:
200: ! define como as mídias são apresentadas
201: !-----+
202:
203: <descriptorBase>
204:   <descriptor id="dVideo1" region="rgVideo1" />
205:   <descriptor id="dLegenda" region="rgLegenda" />
206: </descriptorBase>
207:
208: <!--+-----+
209: ! BASE DE CONECTORES:
210: ! definem o comportamento dos elos
211: !-----+
212:
213: <connectorBase>
214:   <importBase alias="connBase" documentURI="maestroConnectorBase.conn"/>
215: </connectorBase>
216:
217: </head>
218:
219: <!--+-----+
220: ! CORPO DO PROGRAMA:
221: ! define as mídias e estrutura do programa
222: !-----+
223:
224: <body>
225:
226: <!--+-----+
227: ! PONTO DE ENTRADA:
228: ! indica o componente onde o programa inicia
229: !-----+
230:
231: <port id="pInício" component="video1" />
232:
233: <!--+-----+
234: ! MÍDIAS:
235: ! define o local dos arquivos de mídia e as associa com seus descritores

```

```

236: !+++++-----+
237:
238: <media type="video/mpeg" id="video1" src="media/video1.mpg" descriptor="dVideo1">
239:   <!-- âncoras no vídeo que devem ser sincronizadas com a legenda -->
240:   <area id="aVideoLegenda01" begin="5s" end="9s"/>
241:   <area id="aVideoLegenda02" begin="10s" end="14s"/>
242:   <area id="aVideoLegenda03" begin="15s" end="19s"/>
243: </media>
244:
245: <media type="text/html" id="legenda01" src="media/legendas.html#legenda01"
246:   descriptor="dLegenda" />
247: <media type="text/html" id="legenda02" src="media/legendas.html#legenda02"
248:   descriptor="dLegenda" />
249: <!-------+
250: ! ELOS
251: ! define os elos que regem o sincronismo entre as mídias
252: !-----+
253:
254: <!-- início da legenda 01 -->
255:
256: <link id="ILegenda01_start" xconnector="connBase#onBeginStart">
257:   <bind component="video1" interface="aVideoLegenda01" role="onBegin" />
258:   <bind component="legenda01" role="start" />
259: </link>
260:
261: <!-- término da legenda 01 -->
262:
263: <link id="ILegenda01_stop" xconnector="connBase#onEndStop">
264:   <bind component="video1" interface="aVideoLegenda01" role="onEnd" />
265:   <bind component="legenda01" role="stop" />
266: </link>
267:
268: <!-- início da legenda 02 -->
269:
270: <link id="ILegenda02_start" xconnector="connBase#onBeginStart">
271:   <bind component="video1" interface="aVideoLegenda02" role="onBegin" />
272:   <bind component="legenda02" role="start" />
273: </link>
274:
275: <!-- término da legenda 02 -->
276:
277: <link id="ILegenda02_stop" xconnector="connBase#onEndStop">
278:   <bind component="video1" interface="aVideoLegenda02" role="onEnd" />
279:   <bind component="legenda02" role="stop" />
280: </link>
281:
282: <!-- início da legenda 03 -->
283:
284: <link id="ILegenda03_start" xconnector="connBase#onBeginStart">
285:   <bind component="video1" interface="aVideoLegenda03" role="onBegin" />
286:   <bind component="legenda03" role="start" />
287: </link>
288:
289: <!-- término da legenda 03 -->
290:
291: <link id="ILegenda03_stop" xconnector="connBase#onEndStop">
292:   <bind component="video1" interface="aVideoLegenda03" role="onEnd" />
293:   <bind component="legenda03" role="stop" />

```

1

```
294:   </link>
295:
296:   </body>
297: </ncl>
```

Exemplo 06 – Exibindo um vídeo em loop até a intervenção do usuário

Observação: Para executar este exemplo, é necessário que haja, no subdiretório *media*, sob o diretório onde o arquivo NCL estiver localizado, as seguintes mídias:

- dois arquivos de vídeo, chamados *video1.mpg* e *video2.mpg*; e
- uma imagem, chamada *botao_verde.gif*, representando o botão verde.

A Listagem 6 apresenta o código do exemplo, e as linhas em negrito indicam aquelas que foram inseridas ou modificadas para este exemplo:

- **①** conector `onKeySelectionStartNStopNAbortN`;
- **②** elo para reiniciar o vídeo *video1*, criando o *loop*;
- **③** elo que aborta a apresentação do vídeo *video1* e inicia o *video2*, quando o botão verde do controle remoto é pressionado.

Listagem 6. Documento NCL para exibir um vídeo em loop e substituí-lo por outro quando o usuário pressionar a tecla verde.

```
1:  <?xml version="1.0" encoding="ISO-8859-1"?>
2:
3:  <!--+++++--+
4:  EXEMPLO 06
5:
6:  Objetivo: Reproduzir um vídeo em loop, até que o usuário
7:  pressione a tecla verde para mover para o próximo
8:  vídeo.
9:
10: Características:
11:
12: - sincronismo: simples (de início e término de mídias)
13: - interação do usuário: interrupção do vídeo para disparar outro
14:
15: Preparação:
16:
17: Para executar este exemplo, é necessário ter as seguintes mídias no subdiretório
18: mídia a partir do caminho do arquivo NCL:
19:
20: 1) arquivo de vídeo chamado video1.mpg, com uma introdução em loop
21: 2) arquivo de vídeo chamado video2.mpg
22: 3) arquivo de imagem chamado botao_verde.gif, indicando a ação do botão verde
23:
24: !+++++--+
25:
26: <!--+++++--+
27: ! CABEÇALHO NCL:
28: ! define as URIs dos esquemas da NCL
```

```

29: !+++++----->
30:
31: <ncl id="exemplo06" 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">
32:
33: <!------->
34: ! CABEÇALHO DO PROGRAMA
35: !+++++----->
36:
37: <head>
38:
39: <!------->
40: ! BASE DE REGIÕES:
41: ! define as regiões na tela onde as mídias são apresentadas
42: !+++++----->
43:
44: <regionBase>
45:   <region id="rgTV" height="576" width="1024">
46:     <region id="rgVideo1" height="480" width="640" top="48" left="192"/>
47:     <region id="rgBotaoVerde" height="28" width="640" top="528" left="192"/>
48:   </region>
49: </regionBase>
50:
51: <!------->
52: ! BASE DE DESCRIPTORES:
53: ! define como as mídias são apresentadas
54: !+++++----->
55:
56: <descriptorBase>
57:   <descriptor id="dVideo1" region="rgVideo1" />
58:   <descriptor id="dBotaoVerde" region="rgBotaoVerde" />
59: </descriptorBase>
60:
61: <!------->
62: ! BASE DE CONECTORES:
63: ! definem o comportamento dos elos
64: !+++++----->
65:
66: <connectorBase>
67:   <importBase alias="connBase" documentURI="maestroConnectorBase.conn"/>
68:
69:   <causalConnector id="onKeySelectionStartNStopNAbortN">
70:     <connectorParam name="keyCode" />
71:     <simpleCondition role="onSelection" key="$keyCode"/>
72:     <compoundAction operator="par">
73:       <simpleAction role="start" max="unbounded" qualifier="par"/>
74:       <simpleAction role="stop" max="unbounded" qualifier="par"/>
75:       <simpleAction role="abort" max="unbounded" qualifier="par"/>
76:     </compoundAction>
77:   </causalConnector>
78:
79: </connectorBase>
80:
81: </head>
82:
83: <!------->
84: ! CORPO DO PROGRAMA:
85: ! define as mídias e estrutura do programa
86: !+++++----->

```

```

87:
88: <body>
89:
90: <!--+-----+
91: ! PONTO DE ENTRADA:
92: ! indica o componente onde o programa inicia
93: !+-----+
94:
95: <port id="pInício" component="video1" />
96:
97: <!--+-----+
98: ! MÍDIAS:
99: ! define o local dos arquivos de mídia e as associa com seus descritores
100: !+-----+
101:
102: <media type="video/mpeg" id="video1" src="media/video1.mpg" descriptor="dVideo1" />
103:
104: <media type="video/mpeg" id="video2" src="media/video2.mpg" descriptor="dVideo1" />
105:
106: <media type="image/gif" id="botaoVerde" src="media/botao_verde.gif" descriptor="dBotaoVerde" />
107:
108: <!--+-----+
109: ! ELOS
110: ! define os elos que regem o sincronismo entre as mídias
111: !+-----+
112:
113: <!-- início do video1 deve exibir botões -->
114: <link id="!Video1Init" xconnector="connBase#onBeginStart">
115:   <bind component="video1" role="onBegin" />
116:   <bind component="botaoVerde" role="start" />
117: </link>
118:
119: <!-- término do video1 deve dispará-lo novamente (deve tocar em loop) -->
120: <link id="!Video1Loop" xconnector="connBase#onEndStart">
121:   <bind component="video1" role="onEnd" />
122:   <bind component="video1" role="start" />
123: </link>
124:
125: <!-- cancela a exibição do video1 e dispara a exibição do video2 quando a tecla verde é pressionada -->
126: <link id="!SelectBotaoVerde" xconnector="onKeySelectionStartNStopNAbortN">
127:   <bind component="botaoVerde" role="onSelection">
128:     <bndParam name="keyCode" value="GREEN" />
129:   </bind>
130:   <bind component="botaoVerde" role="stop" />
131:   <bnd component="video1" role="abort" />
132:   <bind component="video2" role="start" />
133: </link>
134:
135: </body>
136: </ncl>

```

2

3

Exemplo 07 – Redimensionando uma região de vídeo durante a exibição de uma imagem

Observação: Para executar este exemplo, é necessário que haja, no subdiretório *media* sob o diretório onde o arquivo NCL estiver localizado, as seguintes mídias:

- um vídeo chamado *video1.mpg*; e
- uma imagem chamada *imagem1.gif*.

A listagem a seguir apresenta o código necessário para redimensionar o vídeo quando uma imagem aparecer, a 3 segundos do início do vídeo, e para restaurar o tamanho original, a 8 segundos do vídeo. São criados ① novos conectores, ② a âncora de propriedade que deve ser manipulada, e ③ ④ os elos que a manipulam.

Listagem 7. Documento NCL para redimensionamento de um vídeo enquanto uma imagem está sendo exibida.

```
1: <?xml version="1.0" encoding="ISO-8859-1"?>
2:
3: <!-----+
4: EXEMPLO 07
5:
6: Objetivo: Reproduzir um vídeo numa região da tela e redimensioná-lo num momento
7: de sincronização com uma imagem
8:
9: Características:
10:
11: - sincronismo: simples (de início e término de uma mídia)
12: - interação do usuário: nenhuma
13:
14: Preparação:
15:
16: Para executar este exemplo, é necessário ter as seguintes mídias no subdiretório
17: mídia a partir do caminho do arquivo NCL:
18:
19: 1) arquivo de vídeo, chamado video1.mpg
20: 2) arquivo de imagem, chamado imagem1.gif
21:
22: !-----+
23:
24: <!-----+
25: ! CABEÇALHO NCL:
26: ! define as URIs dos esquemas da NCL
27: !-----+
28:
29: <ncl id="exemplo07"
30: xmlns="http://www.ncl.org.br/NCL3.0/EDTVProfile"
31: xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
32: xsi:schemaLocation="http://www.ncl.org.br/NCL3.0/EDTVProfile
33: http://www.ncl.org.br/NCL3.0/profiles/NCL30EDTV.xsd">
34: <!-----+
35: ! CABEÇALHO DO PROGRAMA
36: !-----+
```

```

37:
38: <head>
39:
40: <!--+-----+
41: ! BASE DE REGIÕES:
42: ! define as regiões na tela onde as mídias são apresentadas
43: !+-----+
44:
45: <regionBase>
46:   <region id="rgTV" height="576" width="1024">
47:     <region id="rgVideo1" height="100%" width="100%" top="0" left="0" />
48:     <region id="rgImagem1" height="50%" width="50%" right="0" bottom="0" />
49:   </region>
50: </regionBase>
51:
52: <!--+-----+
53: ! BASE DE DESCRIPTORES:
54: ! define como as mídias são apresentadas
55: !+-----+
56:
57: <descriptorBase>
58:   <descriptor id="dVideo1" region="rgVideo1" />
59:   <descriptor id="dImagen1" region="rgImagen1" />
60: </descriptorBase>
61:
62: <!--+-----+
63: ! BASE DE CONECTORES:
64: ! definem o comportamento dos elos
65: !+-----+
66:
67: <connectorBase>
68:   <causalConnector id="onBeginSetStartN">
69:     <connectorParam name="var"/>
70:     <simpleCondition role="onBegin"/>
71:     <compoundAction operator="par">
72:       <simpleAction role="set" value="$var"/>
73:       <simpleAction role="start" max="unbounded" qualifier="par"/>
74:     </compoundAction>
75:   </causalConnector>
76:
77:   <causalConnector id="onEndSetStopN">
78:     <connectorParam name="var"/>
79:     <simpleCondition role="onEnd"/>
80:     <compoundAction operator="par">
81:       <simpleAction role="set" value="$var"/>
82:       <simpleAction role="stop" max="unbounded" qualifier="par"/>
83:     </compoundAction>
84:   </causalConnector>
85: </connectorBase>
86:
87: </head>
88:
89: <!--+-----+
90: ! CORPO DO PROGRAMA:
91: ! define as mídias e estrutura do programa
92: !+-----+
93:
94: <body>
95:
96: <!--+-----+

```

1

```

97: ! PONTO DE ENTRADA:
98: ! indica o componente onde o programa inicia
99: !+++++----->
100:
101: <port id="pInício" component="video1" />
102:
103: <!--+-----+
104: ! MÍDIAS:
105: ! define o local dos arquivos de mídia e as associa com seus descritores
106: !+++++----->
107:
108: <media type="video/mpeg" id="video1" src="media/video1.mpg" descriptor="dVideo1">
109:
110: <!-- âncora no vídeo que deve sincronizar a imagem -->
111:
112: <area id="aVideo1Imagem1" begin="3s" end="8s"/>
113:
114: <!-- propriedade que será manipulada pelos elos -->
115:
116: <property name="bounds" />
117:
118: </media>
119:
120: <media type="image/gif" id="imagem1" src="media/imagem1.gif" descriptor="dImagen1" />
121:
122: <!--+-----+
123: ! ELOS
124: ! define os elos que regem o sincronismo entre as mídias
125: !+++++----->
126:
127: <!-- âncora do vídeo 1 deve iniciar a exibição da imagem -->
128:
129: <link id="Video1Imagen1_start" xconnector="onBeginSetStartN">
130:
131:   <bind component="video1" interface="aVideo1Imagen1" role="onBegin" />
132:
133:   <bind component="video1" interface="bounds" role="set">
134:     <bindParam name="var" value="0,0,50%,50%" />
135:     <!-- As dimensões de bounds aparecem na seguinte ordem: left, top, width, height -->
136:   </bind>
137:
138:   <bind component="imagem1" role="start" />
139: </link>
140:
141:
142: <!-- âncora do vídeo 1 deve terminar a exibição da imagem -->
143:
144: <link id="Video1Imagen1_stop" xconnector="onEndSetStopN">
145:
146:   <bind component="video1" interface="aVideo1Imagen1" role="onEnd" />
147:
148:   <bind component="video1" interface="bounds" role="set">
149:     <bindParam name="var" value="0,0,100%,100%" />
150:   </bind>
151:
152:   <bind component="imagem1" role="stop" />
153: </link>
154:
155: </body>
156: </ncl>

```

2

3

4

Exemplo 08 – Trocando um objeto de mídia em resposta a uma ação do usuário

Observação: Para executar este exemplo, é necessário que haja, no subdiretório *media*, sob o diretório onde o arquivo NCL estiver localizado, as seguintes mídias:

- dois arquivos de vídeo, chamados *video1.mpg* e *video2.mpg*; e
- duas imagens, chamadas *botao_vermelho.gif* e *botao_verde.gif*, representando os botões vermelho e verde, respectivamente.

Como as duas mídias devem tocar ao mesmo tempo mas com parâmetros de visibilidade e som distintos, ① um novo descritor para o segundo vídeo (*video2*, linha 107) é criado (*dVideo2*, linha 62) reusando a mesma região do *video1* mas com os valores das propriedades correspondentes definidos no seu descritor (linhas 63 e 64) para que não haja som nem exibição no início.

Para que seja possível alterar as propriedades **visible** e **soundLevel**, é necessário que sejam definidas ② âncoras de propriedades nos nós de mídia correspondentes (linhas 103-104 e 108-109).

Finalmente, é necessário criar dois elos ③ ④ para efetuar a seleção do vídeo, através do conector **onKeySelectionSetN** (linhas 140 a 178). Esse conector é utilizado para trocar as propriedades de visibilidade e volume do som, trocando o vídeo sendo exibido conforme o botão selecionado: o botão vermelho ativa o *video2* e o botão verde ativa o *video1*. Observa-se que, como o elo deve alterar algumas propriedades dos nós de vídeo, cada elemento de ligação (**bind**) deve especificar não apenas o componente de destino do elo (atributo **component**), mas também a propriedade que será modificada (atributo **interface**), tal como definida no nó da mídia. Além disso, como o valor dessa propriedade é modificado, o novo valor deve ser passado como parâmetro (através do elemento **bindParam**):

```
<bind component="video1" interface="aVideo1Visible" role="set">
  <bindParam name="var" value="false" />
</bind>
<bind component="video1" interface="aVideo1SoundLevel" role="set">
  <bindParam name="var" value="0" />
</bind>
```

A Listagem 8 apresenta o código do exemplo, e as linhas em negrito indicam aquelas que foram inseridas ou modificadas para este exemplo.

Listagem 8. Documento NCL com elos para troca de nós de vídeo através da interação do usuário.

```

1:  <?xml version="1.0" encoding="ISO-8859-1"?>
2:
3:  <!--#####
4:      EXEMPLO 08
5:
6:      Objetivo: Reproduzir um vídeo e permitir que o usuário alterne entre dois vídeos,
7:                  numa mesma região da tela, através do pressionamento das teclas
8:                  vermelha e verde do controle remoto. Oculta a imagem correspondente ao
```

```
9:         botão que não fizer efeito no momento.  
10:  
11:     Características:  
12:  
13:         - sincronismo: ponto de alternância entre os vídeos  
14:         - interação do usuário: seleção do vídeo a ser exibido  
15:  
16:     Preparação:  
17:  
18:     Para executar este exemplo, é necessário ter as seguintes mídias no subdiretório  
19:     mídia a partir do caminho do arquivo NCL:  
20:  
21:     1) arquivo de vídeo chamado video1.mpg  
22:     2) arquivo de vídeo chamado video2.mpg  
23:     3) arquivo de imagem chamado botao_vermelho.gif, indicando a ação do botão vermelho  
24:     4) arquivo de imagem chamado botao_verde.gif, indicando a ação do botão verde  
25:  
26: !+++++>  
27:  
28: <!--+++++  
29: ! CABEÇALHO NCL:  
30: ! define as URIs dos esquemas da NCL  
31: !+++++>  
32:  
33: <ncl id="exemplo08" 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">  
34:  
35: <!--+++++  
36: ! CABEÇALHO DO PROGRAMA  
37: !+++++>  
38:  
39: <head>  
40:  
41: <!--+++++  
42: ! BASE DE REGIÕES:  
43: ! define as regiões na tela onde as mídias são apresentadas  
44: !+++++>  
45:  
46: <regionBase>  
47:     <region id="rgTV" height="576" width="1024">  
48:         <region id="rgVideo1" height="480" width="640" top="48" left="192"/>  
49:         <region id="rgBotaoVermelho" height="25" width="25" top="528" left="192"/>  
50:         <region id="rgBotaoVerde" height="25" width="25" top="528" left="242"/>  
51:     </region>  
52: </regionBase>  
53:  
54: <!--+++++  
55: ! BASE DE DESCRIPTORES:  
56: ! define como as mídias são apresentadas  
57: !+++++>  
58:  
59: <descriptorBase>  
60:     <descriptor id="dVideo1" region="rgVideo1" />  
61:  
62:     <descriptor id="dVideo2" region="rgVideo1">  
63:         <descriptorParam name="visible" value="false" />  
64:         <descriptorParam name="soundLevel" value="0"/>  
65:     </descriptor>  
66:
```

```

67: <descriptor id="dBotaoVermelho" region="rgBotaoVermelho" />
68:
69: <descriptor id="dBotaoVerde" region="rgBotaoVerde" />
70: </descriptorBase>
71:
72: <!--+-----+
73: ! BASE DE CONECTORES:
74: ! definem o comportamento dos elos
75: !+-----+
76:
77: <connectorBase>
78:   <importBase alias="connBase" documentURI="maestroConnectorBase.conn"/>
79: </connectorBase>
80:
81: </head>
82:
83: <!--+-----+
84: ! CORPO DO PROGRAMA:
85: ! define as mídias e estrutura do programa
86: !+-----+
87:
88: <body>
89:
90: <!--+-----+
91: ! PONTO DE ENTRADA:
92: ! indica o componente onde o programa inicia
93: !+-----+
94:
95: <port id="pInício" component="video1" />
96:
97: <!--+-----+
98: ! MÍDIAS:
99: ! define o local dos arquivos de mídia e as associa com seus descritores
100: !+-----+
101:
102: <media type="video/mpeg" id="video1" src="media/video1.mpg" descriptor="dVideo1">
103:   <property name="visible" />
104:   <property name="soundLevel" />
105: </media>
106:
107: <media type="video/mpeg" id="video2" src="media/video2.mpg" descriptor="dVideo2">
108:   <property name="visible" />
109:   <property name="soundLevel" />
110: </media>
111:
112: <media type="image/gif" id="botaoVermelho" src="media/botao_vermelho.gif" descriptor="dBotaoVermelho"
113:   />
114: <media type="image/gif" id="botaoVerde" src="media/botao_verde.gif" descriptor="dBotaoVerde" />
115:
116: <!--+-----+
117: ! ELOS
118: ! define os elos que regem o sincronismo entre as mídias
119: !+-----+
120: <!-- início do video1 deve disparar a exibição do video2 e das imagens dos botões -->
121:
122: <link id="lVideo_Botoes_start" xconnector="connBase#onBeginStartN">
123:   <bind component="video1" role="onBegin" />
124:   <bind component="video2" role="start" />
125:   <bind component="botaoVermelho" role="start" />

```

```
126:   <bind component="botaoVerde" role="start" />
127: </link>
128:
129:
130: <!-- término do video1 deve terminar a exibição das imagens dos botões -->
131:
132: <link id="IVideo_Botoes_stop" xconnector="connBase#onEndStopN">
133:   <bind component="video1" role="onEnd" />
134:   <bind component="video2" role="stop" />
135:   <bind component="botaoVerde" role="stop" />
136:   <bind component="botaoVermelho" role="stop" />
137: </link>
138:
139:
140: <!-- alterna para vídeo 2 caso a tecla vermelha seja acionada -->
141:
142: <link id="ISelecionaVideo2" xconnector="connBase#onKeySelectionSetN">
143:   <bind component="botaoVermelho" role="onSelection">
144:     <bindParam name="keyCode" value="RED"/>
145:   </bind>
146:   <bind component="video1" interface="visible" role="set">
147:     <bindParam name="var" value="false"/>
148:   </bind>
149:   <bind component="video1" interface="soundLevel" role="set">
150:     <bindParam name="var" value="0"/>
151:   </bind>
152:   <bind component="video2" interface="visible" role="set">
153:     <bindParam name="var" value="true"/>
154:   </bind>
155:   <bind component="video2" interface="soundLevel" role="set">
156:     <bindParam name="var" value="1"/>
157:   </bind>
158: </link>
159:
160: <!-- alterna para vídeo 1 caso a tecla verde seja acionada -->
161:
162: <link id="ISelecionaVideo1" xconnector="connBase#onKeySelectionSetN">
163:   <bind component="botaoVerde" role="onSelection">
164:     <bindParam name="keyCode" value="GREEN"/>
165:   </bind>
166:   <bind component="video2" interface="visible" role="set">
167:     <bindParam name="var" value="false"/>
168:   </bind>
169:   <bind component="video2" interface="soundLevel" role="set">
170:     <bindParam name="var" value="0"/>
171:   </bind>
172:   <bind component="video1" interface="visible" role="set">
173:     <bindParam name="var" value="true"/>
174:   </bind>
175:   <bind component="video1" interface="soundLevel" role="set">
176:     <bindParam name="var" value="1"/>
177:   </bind>
178: </link>
179:
180: </body>
181:</ncl>
```

3

4

Exemplo 09 – Alternando imagens para identificar ações disponíveis

Observação: Para executar este exemplo, é necessário que haja, no subdiretório *media*, sob o diretório onde o arquivo NCL estiver localizado, as seguintes mídias:

- dois arquivos de vídeo, chamados *video1.mpg* e *video2.mpg*; e
- duas imagens, chamadas *botao_vermelho.gif* e *botao_verde.gif*, representando os botões vermelho e verde, respectivamente.

Em primeiro lugar, o elo de exibição inicial do vídeo e dos botões não deve mais exibir o botão verde, pois pressionar a tecla verde enquanto o vídeo *video1* está tocando não produz efeito (item removido no elo definido nas linhas 123 a 127 ❶). Além disto, a única alteração a ser feita é a inclusão de dois elos ❷, que alternam a exibição das imagens correspondentes aos botões (linhas 179 a 195). Esses elos utilizam o conector **onKeySelectionStartNStopN**, especificado para modificar a exibição de um ou mais nós de mídia conforme o pressionamento de uma determinada tecla.

A Listagem 9 apresenta o código do exemplo, e as linhas em negrito indicam aquelas que foram inseridas ou modificadas para este exemplo.

Listagem 9. Documento NCL com elos para alternar a exibição de imagens através da interação do usuário para refletir a alternância entre nós de vídeo e as oportunidades de interação.

```
1: <?xml version="1.0" encoding="ISO-8859-1"?>
2:
3: <!-----+
4: EXEMPLO 09
5:
6: Objetivo: Reproduzir um vídeo e permitir que o usuário alterne entre dois vídeos,
7: numa mesma região da tela, através do pressionamento das teclas
8: vermelha e verde do controle remoto. Oculta a imagem correspondente ao
9: botão que não fizer efeito no momento.
10:
11: Características:
12:
13: - sincronismo: ponto de alternância entre os vídeos
14: - interação do usuário: seleção do vídeo a ser exibido
15:
16: Preparação:
17:
18: Para executar este exemplo, é necessário ter as seguintes mídias no subdiretório
19: mídia a partir do caminho do arquivo NCL:
20:
21: 1) arquivo de vídeo chamado video1.mpg
22: 2) arquivo de vídeo chamado video2.mpg
23: 3) arquivo de imagem chamado botao_vermelho.gif, indicando a ação do botão vermelho
24: 4) arquivo de imagem chamado botao_verde.gif, indicando a ação do botão verde
25:
26: !-----+
27:
28: <!-----+
29: ! CABEÇALHO NCL:
```

```
30: ! define as URLs dos esquemas da NCL
31: !+++++----->
32:
33: <ncl id="exemplo09" 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">
34:
35: <!------->
36: ! CABEÇALHO DO PROGRAMA
37: !+++++----->
38:
39: <head>
40:
41: <!------->
42: ! BASE DE REGIÕES:
43: ! define as regiões na tela onde as mídias são apresentadas
44: !+++++----->
45:
46: <regionBase>
47:   <region id="rgTV" height="576" width="1024">
48:     <region id="rgVideo1" height="480" width="640" top="48" left="192"/>
49:     <region id="rgBotaoVermelho" height="25" width="25" top="528" left="192"/>
50:     <region id="rgBotaoVerde" height="25" width="25" top="528" left="242"/>
51:   </region>
52: </regionBase>
53:
54: <!------->
55: ! BASE DE DESCRIPTORES:
56: ! define como as mídias são apresentadas
57: !+++++----->
58:
59: <descriptorBase>
60:   <descriptor id="dVideo1" region="rgVideo1" />
61:
62:   <descriptor id="dVideo2" region="rgVideo1">
63:     <descriptorParam name="visible" value="false" />
64:     <descriptorParam name="soundLevel" value="0"/>
65:   </descriptor>
66:
67:   <descriptor id="dBotaoVermelho" region="rgBotaoVermelho" />
68:
69:   <descriptor id="dBotaoVerde" region="rgBotaoVerde" />
70:
71: </descriptorBase>
72:
73: <!------->
74: ! BASE DE CONECTORES:
75: ! definem o comportamento dos elos
76: !+++++----->
77:
78: <connectorBase>
79:   <importBase alias="connBase" documentURI="maestroConnectorBase.conn"/>
80: </connectorBase>
81:
82: </head>
83:
84: <!------->
85: ! CORPO DO PROGRAMA:
86: ! define as mídias e estrutura do programa
87: !+++++----->
```

```

88:
89: <body>
90:
91: <!--+-----+-----+-----+-----+-----+-----+-----+-----+
92: ! PONTO DE ENTRADA:
93: ! indica o componente onde o programa inicia
94: !+-----+-----+-----+-----+-----+-----+-----+----->
95:
96: <port id="pInício" component="video1" />
97:
98: <!--+-----+-----+-----+-----+-----+-----+-----+-----+
99: ! MÍDIAS:
100: ! define o local dos arquivos de mídia e as associa com seus descritores
101: !+-----+-----+-----+-----+-----+-----+-----+----->
102:
103: <media type="video/mpeg" id="video1" src="media/video1.mpg" descriptor="dVideo1">
104:   <property name="visible" />
105:   <property name="soundLevel" />
106: </media>
107:
108: <media type="video/mpeg" id="video2" src="media/video2.mpg" descriptor="dVideo2">
109:   <property name="visible" />
110:   <property name="soundLevel" />
111: </media>
112:
113: <media type="image/gif" id="botaoVermelho" src="media/botao_vermelho.gif" descriptor="dBotaoVermelho" />
114: <media type="image/gif" id="botaoVerde" src="media/botao_verde.gif" descriptor="dBotaoVerde" />
115:
116: <!--+-----+-----+-----+-----+-----+-----+-----+-----+
117: ! ELOS
118: ! define os elos que regem o sincronismo entre as mídias
119: !+-----+-----+-----+-----+-----+-----+-----+----->
120:
121: <!-- iníco do video1 deve disparar a exibição do video2 e das imagens do botão vermelho -->
122:
123: <link id="IVideo_Botoes_start" xconnector="connBase#onBeginStartN">
124:   <bind component="video1" role="onBegin" />
125:   <bind component="video2" role="start" />
126:   <bind component="botaoVermelho" role="start" />
127: </link>
128:
129: <!-- término do video1 deve terminar a exibição das imagens dos botões -->
130:
131: <link id="IVideo_Botoes_stop" xconnector="connBase#onEndStopN">
132:   <bind component="video1" role="onEnd" />
133:   <bind component="video2" role="stop" />
134:   <bind component="botaoVerde" role="stop" />
135:   <bind component="botaoVermelho" role="stop" />
136: </link>
137:
138: <!-- alterna para vídeo 2 caso a tecla vermelha seja acionada -->
139:
140: <link id="ISelecionaVideo2" xconnector="connBase#onKeySelectionSetN">
141:   <linkParam name="keyCode" value="RED" />
142:   <bind component="botaoVermelho" role="onSelection" />
143:   <bind component="video1" interface="visible" role="set">
144:     <bindParam name="var" value="false" />
145:   </bind>
146:   <bind component="video1" interface="soundLevel" role="set">
147:     <bindParam name="var" value="0" />

```

```

148: </bind>
149:
150: <bind component="video2" interface="visible" role="set">
151:   <bindParam name="var" value="true" />
152: </bind>
153: <bind component="video2" interface="soundLevel" role="set">
154:   <bindParam name="var" value="1" />
155: </bind>
156: </link>
157:
158: <!-- alterna para vídeo 1 caso a tecla verde seja acionada -->
159:
160: <link id="ISeleccionaVideo1" xconnector="connBase#onKeySelectionSetN">
161:   <linkParam name="keyCode" value="GREEN" />
162:   <bind component="botaoVerde" role="onSelection" />
163:
164:   <bind component="video2" interface="visible" role="set">
165:     <bindParam name="var" value="false" />
166:   </bind>
167:   <bind component="video2" interface="soundLevel" role="set">
168:     <bindParam name="var" value="0" />
169:   </bind>
170:
171: <bind component="video1" interface="visible" role="set">
172:   <bindParam name="var" value="true" />
173: </bind>
174: <bind component="video1" interface="soundLevel" role="set">
175:   <bindParam name="var" value="1" />
176: </bind>
177: </link>
178:
179: <!-- oculta a tecla vermelha quando ela é pressionada -->
180: <link id="IToggleBotaoVermelho" xconnector="connBase#onKeySelectionStartNStopN">
181:   <bind component="botaoVermelho" role="onSelection">
182:     <bindParam name="keyCode" value="RED" />
183:   </bind>
184:   <bind component="botaoVerde" role="start" />
185:   <bind component="botaoVermelho" role="stop" />
186: </link>
187:
188: <!-- oculta a tecla verde quando ela é pressionada -->
189: <link id="IToggleBotaoVerde" xconnector="connBase#onKeySelectionStartNStopN">
190:   <bind component="botaoVerde" role="onSelection">
191:     <bindParam name="keyCode" value="GREEN" />
192:   </bind>
193:   <bind component="botaoVerde" role="stop" />
194:   <bind component="botaoVermelho" role="start" />
195: </link>
196:
197: </body>
198: </ncl>

```

2

Exemplo 10 – Simulação de um menu de DVD

Observação: Para executar este exemplo, é necessário que haja, no subdiretório *media*, sob o diretório onde o arquivo NCL estiver localizado, as seguintes mídias:

- três arquivos de vídeo, chamados video1.mpg, video2.mpg e video3.mpg;
- duas imagens, chamadas botao_vermelho.gif e botao_verde.gif, representando os botões vermelho e verde, respectivamente; e
- dois arquivos de áudio, chamados audioEn.mp3 e audioPt.mp3, cada qual num idioma (*en* = inglês e *pt* = português).

Para armazenar a opção selecionada, será utilizado um **④** nó do tipo **settings** com uma propriedade *idioma* (linhas 121 a 123). Já para selecionar o nó de áudio a ser reproduzido ao final do *video2*, são definidas **② rules** (regras, linhas 78 a 81) e **⑥** um **switch** (linhas 151 a 157).

Quando o botão vermelho ou o verde é selecionado, o idioma correspondente é armazenado na propriedade *idioma*, através do conector **③ OnKeySelectionSetNStartNStopNAbortN** utilizado pelos elos **⑦ lSelectBotaoVermelhoIdioma** e **lSelectBotaoVerdeIdioma** (linhas 193 a 219). Este mesmo conector é o responsável por interromper o *video1* e iniciar a exibição do *video2*. Quando o *video2* termina, o elo **lVideo2StartVideo3**, através do conector **OnEndStart**, ativa **⑤** o nó de contexto *contextoFilme* (linhas 221 a 225). A porta desse contexto ativa o *video3* (linha 141), que por sua vez dispara, sincronamente, através do conector **onBeginStart** (linhas 161 a 164), **⑥** o **switch** *switchAudioIdioma* (linhas 151 a 157). Este switch é regido pelas regras *rEn* e *rPt* (linhas 79 e 80), definidas previamente na seção **② ruleBase** (linhas 78 a 81). Nota-se que foi criado um segundo **①** descritor de vídeo para o *video3* (linhas 64 a 66), pois ele deve ser exibido sem som (*soundLevel=0*).

A Listagem 10 apresenta o código do exemplo, e as linhas em negrito indicam aquelas que foram inseridas ou modificadas para este exemplo.

Listagem 10. Documento NCL para exibir um vídeo em loop e, mediante a seleção do usuário, substituí-lo por um segundo vídeo e, finalmente, exibir um terceiro vídeo e um áudio conforme a seleção de idioma.

```

1:  <?xml version="1.0" encoding="ISO-8859-1"?>
2:
3:  <!-----+
4:  EXEMPLO 10
5:
6:  Objetivo: Reproduzir um vídeo e permitir que o usuário selecione
7:          o idioma do áudio, através do pressionamento das teclas
8:          vermelha e verde do controle remoto.
9:
10: Características:
11:
12: - sincronismo: simples (de início e término de mídias)
13: - interação do usuário: seleção do áudio a ser exibido
14:
15: Preparação:
16:
17: Para executar este exemplo, é necessário ter as seguintes mídias no subdiretório
18: mídia a partir do caminho do arquivo NCL:
19:
20: 1) arquivo de vídeo chamado video1.mpg, com uma introdução em loop

```

```

21: 2) arquivo de vídeo chamado video2.mpg, com uma abertura do produtor do vídeo
22: 3) arquivo de vídeo chamado video3.mpg, que contém o filme em si
23: 4) arquivo de áudio chamado audio1.mp3, com o áudio em inglês
24: 5) arquivo de áudio chamado audio2.mp3, com o áudio em português
25: 6) arquivo de imagem chamado botao_vermelho.gif, indicando a ação do botão vermelho
26: 7) arquivo de imagem chamado botao_verde.gif, indicando a ação do botão verde
27:
28: !+++++>
29:
30: <!--+++++>
31: ! CABEÇALHO NCL:
32: ! define as URIs dos esquemas da NCL
33: !+++++>
34:
35: <ncl id="exemplo10" 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">
36:
37: <!--+++++>
38: ! CABEÇALHO DO PROGRAMA
39: !+++++>
40:
41: <head>
42:
43: <!--+++++>
44: ! BASE DE REGIÕES:
45: ! define as regiões na tela onde as mídias são apresentadas
46: !+++++>
47:
48: <regionBase>
49:   <region id="rgTV" height="576" width="1024">
50:     <region id="rgVideo1" height="480" width="640" top="48" left="192"/>
51:     <region id="rgBotaoVermelho" height="25" width="25" top="528" left="192"/>
52:     <region id="rgBotaoVerde" height="25" width="25" top="528" left="242"/>
53:   </region>
54: </regionBase>
55:
56: <!--+++++>
57: ! BASE DE DESCRIPTORES:
58: ! define como as mídias são apresentadas
59: !+++++>
60:
61: <descriptorBase>
62:   <descriptor id="dVideo1" region="rgVideo1" />
63:
64:   <descriptor id="dVideo3" region="rgVideo1">
65:     <descriptorParam name="soundLevel" value="0" />
66:   </descriptor>
67:   <descriptor id="dAudio1" />
68:
69:   <descriptor id="dBotaoVermelho" region="rgBotaoVermelho" />
70:   <descriptor id="dBotaoVerde" region="rgBotaoVerde" />
71: </descriptorBase>
72:
73: <!--+++++>
74: ! BASE DE REGRAS:
75: ! definem o comportamento dos elos
76: !+++++>
77:
78: <ruleBase>

```

1

2

```

79:   <rule id="rEn" var="idioma" comparator="eq" value="en" />
80:   <rule id="rPt" var="idioma" comparator="eq" value="pt" />
81: </ruleBase>
82:
83: <!-----+
84: ! BASE DE CONECTORES:
85: ! definem o comportamento dos elos
86: !+-----+
87:
88: <connectorBase>
89:   <importBase alias="connBase" documentURI="maestroConnectorBase.conn"/>
90:
91:   <causalConnector id="onKeySelectionSetNStartNStopNAbortN">
92:     <connectorParam name="keyCode" />
93:     <connectorParam name="var" />
94:     <simpleCondition role="onSelection" key="$keyCode"/>
95:     <compoundAction operator="par">
96:       <simpleAction role="set" value="$var" max="unbounded" qualifier="par"/>
97:       <simpleAction role="start" max="unbounded" qualifier="par"/>
98:       <simpleAction role="stop" max="unbounded" qualifier="par"/>
99:       <simpleAction role="abort" max="unbounded" qualifier="par"/>
100:    </compoundAction>
101:   </causalConnector>
102:
103: </connectorBase>
104:
105: </head>
106:
107: <!-----+
108: ! CORPO DO PROGRAMA:
109: ! define as mídias e estrutura do programa
110: !+-----+
111:
112: <body>
113:
114: <!-----+
115: ! PONTO DE ENTRADA:
116: ! indica o componente onde o programa inicia
117: !+-----+
118:
119: <port id="plnicio" component="video1" />
120:
121: <media type="application/x-ginga-settings" id="nodeSettings">
122:   <property name="idioma" />
123: </media>
124:
125: <!-----+
126: ! MÍDIAS:
127: ! define o local dos arquivos de mídia e as associa com seus descritores
128: !+-----+
129: <media type="video/mpeg" id="video1" src="media/video1.mpg" descriptor="dVideo1" />
130:
131: <media type="video/mpeg" id="video2" src="media/video2.mpg" descriptor="dVideo1" />
132:
133: <media type="image/gif" id="botaoVermelho" src="media/botao_vermelho.gif" descriptor="dBotaoVermelho" />
134: <media type="image/gif" id="botaoVerde" src="media/botao_verde.gif" descriptor="dBotaoVerde" />
135:
136: <!-----+
137: ! CONTEXTO:

```

3

4

```

138: ! contém o vídeo do filme e do switch com os áudios
139: !+++++-----+
140: <context id="contextoFilme">
141:   <port id="pVideo3" component="video3" />
142:
143:   <media type="video/mpeg" id="video3" src="media/video3.mpg" descriptor="dVideo3" />
144:
145:   <!--+++++-----+
146:   ! SWITCH: contexto que
147:   ! contém os áudios, com regra de seleção entre eles
148:   ! conforme o idioma
149:   !+++++-----+
150:
151: <switch id="switchAudiodioma">
152:   <bindRule rule="rEn" constituent="audioEn" />
153:   <bindRule rule="rPt" constituent="audioPt" />
154:
155:   <media type="audio/mp3" id="audioEn" src="media/audioEn.mp3" descriptor="dAudio1" />
156:   <media type="audio/mp3" id="audioPt" src="media/audioPt.mp3" descriptor="dAudio1" />
157: </switch>
158:
159: <!-- início do video3 deve disparar a reprodução do áudio -->
160:
161: <link id="IVideo3AudioStart" xconnector="connBase#onBeginStart">
162:   <bind component="video3" role="onBegin" />
163:   <bind component="switchAudiodioma" role="start" />
164: </link>
165:
166: <!-- término do video3 deve parar a reprodução do áudio -->
167:
168: <link id="IVideo3AudioStop" xconnector="connBase#onEndStop">
169:   <bind component="video3" role="onEnd" />
170:   <bind component="switchAudiodioma" role="stop" />
171: </link>
172:
173: </context>
174:
175: <!--+++++-----+
176:   ! ELOS
177:   ! define os elos que regem o sincronismo entre as mídias
178:   !+++++-----+
179:
180: <!-- início do video1 deve exibir botões -->
181: <link id="IVideo1Init" xconnector="connBase#onBeginStart">
182:   <bind component="video1" role="onBegin" />
183:   <bind component="botaoVermelho" role="start" />
184:   <bind component="botaoVerde" role="start" />
185: </link>
186:
187: <!-- término do video1 deve dispará-lo novamente (deve tocar em loop) -->
188: <link id="IVideo1Loop" xconnector="connBase#onEndStart">
189:   <bind component="video1" role="onEnd" />
190:   <bind component="video1" role="start" />
191: </link>
192:
193: <!-- define idioma inglês quando a tecla vermelha é pressionada -->
194: <link id="ISelectBotaoVermelholdioma" xconnector="onKeySelectionSetNStartNStopNAbortN">
195:   <bind component="botaoVermelho" role="onSelection">
196:     <bindParam name="keyCode" value="RED" />
197:   </bind>

```

5

6

7

```

198:   <bind component="nodeSettings" interface="idioma" role="set">
199:     <bindParam name="var" value="en" />
200:   </bind>
201:   <bind component="botaoVerde" role="stop" />
202:   <bind component="botaoVermelho" role="stop" />
203:   <bind component="video1" role="abort" />
204:   <bind component="video2" role="start" />
205: </link>
206:
207: <!-- define idioma português quando a tecla verde é pressionada -->
208: <link id="ISelectBotaoVerdeIdioma" xconnector="onKeySelectionSetNStartNStopNAbortN">
209:   <bind component="botaoVerde" role="onSelection">
210:     <bindParam name="keyCode" value="GREEN" />
211:   </bind>
212:   <bind component="nodeSettings" interface="idioma" role="set">
213:     <bindParam name="var" value="pt" />
214:   </bind>
215:   <bind component="botaoVerde" role="stop" />
216:   <bind component="botaoVermelho" role="stop" />
217:   <bind component="video1" role="abort" />
218:   <bind component="video2" role="start" />
219: </link>
220:
221: <!-- término do video2 deve disparar o video3 em seu contexto -->
222: <link id="IVideo2StartVideo3" xconnector="connBase#onEndStart">
223:   <bind component="video2" role="onEnd" />
224:   <bind component="contextoFilme" interface="pVideo3" role="start" />
225: </link>
226:
227: </body>
228: </ncl>

```

Exemplo 11 – Simulação de um menu de DVD com feedback

A Listagem 11 apresenta o código NCL das modificações necessárias para esse exemplo: ① o conector com retardo (linhas 93 a 104) e ② os elos que o utilizam (linhas 196 a 222).

Listagem 11. Modificações sobre a listagem anterior para, mediante a seleção do usuário, fornecer feedback momentâneo.

```

1:  <?xml version="1.0" encoding="ISO-8859-1"?>
2:
3:  <!--#####
4:      EXEMPLO 11
5:
6:      Objetivo: Reproduzir um vídeo e permitir que o usuário selecione
7:          o idioma do áudio, através do pressionamento das teclas
8:          vermelha e verde do controle remoto
9:          e com feedback imediato para o usuário.
10:
11:     Características:
12:
13:     - sincronismo: simples (de início e término de mídias)
14:     - interação do usuário: seleção do áudio a ser exibido
15:
16:     Preparação:
17:

```

```

18: Para executar este exemplo, é necessário ter as seguintes mídias no subdiretório
19: mídia a partir do caminho do arquivo NCL:
20:
21: 1) arquivo de vídeo chamado video1.mpg, com uma introdução em loop
22: 2) arquivo de vídeo chamado video2.mpg, com uma abertura do produtor do vídeo
23: 3) arquivo de vídeo chamado video3.mpg, que contém o filme em si
24: 4) arquivo de áudio chamado audio1.mp3, com o áudio em inglês
25: 5) arquivo de áudio chamado audio2.mp3, com o áudio em português
26: 6) arquivo de imagem chamado botao_vermelho.gif, indicando a ação do botão vermelho
27: 7) arquivo de imagem chamado botao_verde.gif, indicando a ação do botão verde
28:
29: !+++++>
30:
31: <!--+
32: ! CABEÇALHO NCL:
33: ! define as URIs dos esquemas da NCL
34: !+++++>
35:
36: <ncl id="exemplo11" 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">
37:
38: <!--+
39: ! CABEÇALHO DO PROGRAMA
40: !+++++>
41:
42: <head>
43:
44: <!--+
45: ! BASE DE REGIÕES:
46: ! define as regiões na tela onde as mídias são apresentadas
47: !+++++>
48:
49: <regionBase>
50:   <region id="rgTV" height="576" width="1024">
51:     <region id="rgVideo1" height="480" width="640" top="48" left="192" />
52:     <region id="rgBotaoVermelho" height="25" width="25" top="528" left="192" />
53:     <region id="rgBotaoVerde" height="25" width="25" top="528" left="242" />
54:   </region>
55: </regionBase>
56:
57: <!--+
58: ! BASE DE DESCRIPTORES:
59: ! define como as mídias são apresentadas
60: !+++++>
61:
62: <descriptorBase>
63:   <descriptor id="dVideo1" region="rgVideo1" />
64:
65:   <descriptor id="dVideo3" region="rgVideo1">
66:     <descriptorParam name="soundLevel" value="0" />
67:   </descriptor>
68:
69:   <descriptor id="dAudio1" />
70:
71:   <descriptor id="dBotaoVermelho" region="rgBotaoVermelho" />
72:   <descriptor id="dBotaoVerde" region="rgBotaoVerde" />
73: </descriptorBase>
74:
75: <!--+

```

```

76: ! BASE DE REGRAS:
77: ! definem o comportamento dos elos
78: !+++++-->
79:
80: <ruleBase>
81:   <rule id="rEn" var="idioma" comparator="eq" value="en" />
82:   <rule id="rPt" var="idioma" comparator="eq" value="pt" />
83: </ruleBase>
84:
85: <!--+++++-->
86: ! BASE DE CONECTORES:
87: ! definem o comportamento dos elos
88: !+++++-->
89:
90: <connectorBase>
91:   <importBase alias="connBase" documentURI="maestroConnectorBase.conn" />
92:
93:   <causalConnector id="onKeySelectionSetNStopNDStartNDStopNDAbortN">
94:     <connectorParam name="keyCode" />
95:     <connectorParam name="var" />
96:     <simpleCondition role="onSelection" key="$keyCode"/>
97:     <compoundAction operator="par">
98:       <simpleAction role="set" value="$var" max="unbounded" qualifier="par"/>
99:       <simpleAction role="stop" max="unbounded" qualifier="par"/>
100:      <simpleAction role="dstart" delay="0.5s" max="unbounded" qualifier="par"/>
101:      <simpleAction role="dstop" delay="0.5s" max="unbounded" qualifier="par"/>
102:      <simpleAction role="dabort" delay="0.5s" max="unbounded" qualifier="par"/>
103:    </compoundAction>
104:  </causalConnector>
105:
106: </connectorBase>
107:
108: </head>
109:
110: <!--+++++-->
111: ! CORPO DO PROGRAMA:
112: ! define as mídias e estrutura do programa
113: !+++++-->
114:
115: <body>
116:
117: <!--+++++-->
118: ! PONTO DE ENTRADA:
119: ! indica o componente onde o programa inicia
120: !+++++-->
121:
122: <port id="plinicio" component="video1" />
123:
124:   <media type="application/x-ginga-settings" id="nodeSettings">
125:     <property name="idioma" />
126:   </media>
127:
128: <!--+++++-->
129: ! MÍDIAS:
130: ! define o local dos arquivos de mídia e as associa com seus descritores
131: !+++++-->
132: <media type="video/mpeg" id="video1" src="media/video1.mpg" descriptor="dVideo3" />
133:
134: <media type="video/mpeg" id="video2" src="media/video2.mpg" descriptor="dVideo1" />
135:

```

1

```

136: <media type="image/gif" id="botaoVermelho" src="media/botao_vermelho.gif" descriptor="dBotaoVermelho"
137:   />
138: <media type="image/gif" id="botaoVerde" src="media/botao_verde.gif" descriptor="dBotaoVerde" />
139: <!--+++++-----+
140: ! CONTEXTO:
141: ! contém o vídeo do filme e do switch com os áudios
142: !+++++-----+
143: <context id="contextoFilme">
144:   <port id="pVideo3" component="video3" />
145:
146:   <media type="video/mpeg" id="video3" src="media/video3.mpg" descriptor="dVideo3" />
147:
148: <!--+++++-----+
149: ! SWITCH: contexto que
150: ! contém os áudios, com regra de seleção entre eles
151: ! conforme o idioma
152: !+++++-----+
153:
154: <switch id="switchAudiodioma">
155:   <bindRule rule="rEn" constituent="audioEn" />
156:   <bindRule rule="rPt" constituent="audioPt" />
157:
158:   <media type="audio/mp3" id="audioEn" src="media/audioEn.mp3" descriptor="dAudio1" />
159:   <media type="audio/mp3" id="audioPt" src="media/audioPt.mp3" descriptor="dAudio1" />
160: </switch>
161:
162: <!-- início do video3 deve disparar a reprodução do áudio -->
163:
164: <link id="lVideo3AudioStart" xconnector="connBase#onBeginStart">
165:   <bind component="video3" role="onBegin" />
166:   <bind component="switchAudiodioma" role="start" />
167: </link>
168:
169: <!-- término do video3 deve parar a reprodução do áudio -->
170:
171: <link id="lVideo3AudioStop" xconnector="connBase#onEndStop">
172:   <bind component="video3" role="onEnd" />
173:   <bind component="switchAudiodioma" role="stop" />
174: </link>
175:
176: </context>
177:
178: <!--+++++-----+
179: ! ELOS
180: ! define os elos que regem o sincronismo entre as mídias
181: !+++++-----+
182:
183: <!-- início do video1 deve exibir botões -->
184: <link id="lVideo1Init" xconnector="connBase#onBeginStart">
185:   <bind component="video1" role="onBegin" />
186:   <bind component="botaoVermelho" role="start" />
187:   <bind component="botaoVerde" role="start" />
188: </link>
189:
190: <!-- término do video1 deve dispará-lo novamente (deve tocar em loop) -->
191: <link id="lVideo1Loop" xconnector="connBase#onEndStart">
192:   <bind component="video1" role="onEnd" />
193:   <bind component="video1" role="start" />
194: </link>

```

```

195:
196: <!-- define idioma inglês quando a tecla vermelha é pressionada -->
197: <link id="ISelectBotaoVermelhol idioma"
198:   xconnector="onKeySelectionSetNStopNDStartNDStopNDAbortN">
199:   <bind component="botaoVermelho" role="onSelection">
200:     <bindParam name="keyCode" value="RED" />
201:   </bind>
202:   <bind component="nodeSettings" interface="idioma" role="set">
203:     <bindParam name="var" value="en" />
204:   </bind>
205:   <bind component="botaoVerde" role="stop" />
206:   <bind component="botaoVermelho" role="dstop" />
207:   <bind component="video1" role="dabort" />
208:   <bind component="video2" role="dstart" />
209: </link>
210: <!-- define idioma português quando a tecla verde é pressionada -->
211: <link id="ISelectBotaoVerdeidioma" xconnector="onKeySelectionSetNStopNDStartNDStopNDAbortN">
212:   <bind component="botaoVerde" role="onSelection">
213:     <bindParam name="keyCode" value="GREEN" />
214:   </bind>
215:   <bind component="nodeSettings" interface="idioma" role="set">
216:     <bindParam name="var" value="pt" />
217:   </bind>
218:   <bind component="botaoVermelho" role="stop" />
219:   <bind component="botaoVerde" role="dstop" />
220:   <bind component="video1" role="dabort" />
221:   <bind component="video2" role="dstart" />
222: </link>
223: <!-- término do video2 deve disparar o video3 em seu contexto -->
224: <link id="IVideo2StartVideo3" xconnector="connBase#onEndStart">
225:   <bind component="video2" role="onEnd" />
226:   <bind component="contextoFilme" interface="pVideo3" role="start" />
227: </link>
228:
229:
230: </body>
231: </ncl>

```

Exemplo 12 – Seleção através das setas do controle remoto

A Listagem 12 apresenta o código do programa que, dada uma seleção, dispara uma página HTML correspondente (*selecao1.html*, *selecao2.html* etc).

Listagem 12. Documento NCL ilustrando um menu com 6 opções.

```

1:  <?xml version="1.0" encoding="ISO-8859-1"?>
2:  <ncl id="exemplo12" xmlns="http://www.ncl.org.br/NCL3.0/EDTVProfile">
3:
4:  <!-------+
5:  ! CABEÇALHO DO PROGRAMA
6:  !-----+
7:
8:  <head>
9:
10: <!-------+
11: ! BASE DE REGIÕES:

```

```

12: ! define as regiões na tela onde as mídias são apresentadas
13: !+++++-----+
14:
15: <regionBase>
16:   <region id="rgTV" width="1080" height="768">
17:     <region id="rgVideo1" left="448" top="100" width="1024" height="768" zIndex="1"/>
18:     <region id="rgOpcao1" left="200" top="100" width="200" height="50" zIndex="2"/>
19:     <region id="rgOpcao2" left="200" top="150" width="200" height="50" zIndex="2"/>
20:     <region id="rgOpcao3" left="200" top="200" width="200" height="50" zIndex="2"/>
21:     <region id="rgOpcao4" left="200" top="250" width="200" height="50" zIndex="2"/>
22:     <region id="rgOpcao5" left="200" top="300" width="200" height="50" zIndex="2"/>
23:     <region id="rgOpcao6" left="200" top="350" width="200" height="50" zIndex="2"/>
24:     <region id="rgResultado" left="150" top="100" width="200" height="100" zIndex="2"/>
25:   </region>
26: </regionBase>
27:
28: <!--+++++-----+
29: ! BASE DE DESCRIPTORES:
30: ! define como as mídias são apresentadas
31: !+++++-----+
32:
33: <descriptorBase>
34:   <descriptor id="dVideo1" region="rgVideo1"/>
35:   <descriptor id="dOpcao1" region="rgOpcao1" focusIndex="1" moveDown="2" focusBorderWidth="-2"
focusBorderColor="blue"/>
36:   <descriptor id="dOpcao2" region="rgOpcao2" focusIndex="2" moveDown="3" moveUp="1"
focusBorderWidth="-2" focusBorderColor="blue"/>
37:   <descriptor id="dOpcao3" region="rgOpcao3" focusIndex="3" moveDown="4" moveUp="2"
focusBorderWidth="-2" focusBorderColor="blue"/>
38:   <descriptor id="dOpcao4" region="rgOpcao4" focusIndex="4" moveDown="5" moveUp="3"
focusBorderWidth="-2" focusBorderColor="blue"/>
39:   <descriptor id="dOpcao5" region="rgOpcao5" focusIndex="5" moveDown="6" moveUp="4"
focusBorderWidth="-2" focusBorderColor="blue"/>
40:   <descriptor id="dOpcao6" region="rgOpcao6" focusIndex="6" moveUp="5" focusBorderWidth="-2"
focusBorderColor="blue"/>
41:   <descriptor id="dResultado" region="rgResultado" />
42: </descriptorBase>
43:
44: <!--+++++-----+
45: ! BASE DE REGRAS:
46: ! definem regras utilizadas em switches para a seleção de nós
47: !+++++-----+
48:
49: <ruleBase>
50:   <rule id="r1" var="opcao" comparator="eq" value="1"/>
51:   <rule id="r2" var="opcao" comparator="eq" value="2"/>
52:   <rule id="r3" var="opcao" comparator="eq" value="3"/>
53:   <rule id="r4" var="opcao" comparator="eq" value="4"/>
54:   <rule id="r5" var="opcao" comparator="eq" value="5"/>
55:   <rule id="r6" var="opcao" comparator="eq" value="6"/>
56: </ruleBase>
57:
58: <!--+++++-----+
59: ! BASE DE CONECTORES:
60: ! definem o comportamento dos elos
61: !+++++-----+
62:
63: <connectorBase>
64:   <importBase alias="connBase" documentURI="maestroConnectorBase.conn" />
65:
66:   <causalConnector id="onSelectionSetNStartNStopN">
```

```

67:      <connectorParam name="var"/>
68:      <simpleCondition role="onSelection"/>
69:      <compoundAction operator="seq">
70:          <simpleAction role="set" value="$var" max="unbounded" qualifier="par"/>
71:          <simpleAction role="stop" max="unbounded" qualifier="par"/>
72:          <simpleAction role="start" max="unbounded" qualifier="par"/>
73:      </compoundAction>
74:  </causalConnector>
75:
76:  </connectorBase>
77:
78: </head>
79:
80: <!-----+
81: ! CORPO DO PROGRAMA:
82: ! define as mídias e estrutura do programa
83: !+-----+
84:
85: <body>
86:
87: <!-----+
88: ! PONTO DE ENTRADA:
89: ! indica o componente onde o programa inicia
90: !+-----+
91:
92: <port id="pEntryPoint" component="video1"/>
93:
94: <media type="application/x-ginga-settings" id="nodeSettings">
95:     <property name="opcao"/>
96: </media>
97:
98: <!-----+
99: ! MÍDIAS:
100: ! define o local dos arquivos de mídia e as associa com seus descritores
101: !+-----+
102:
103: <media id="video1" src="media/video1.mpg" descriptor="dVideo1"/>
104:
105: <media id="opcao1" src="media/opcao1.gif" descriptor="dOpcao1"/>
106: <media id="opcao2" src="media/opcao2.gif" descriptor="dOpcao2"/>
107: <media id="opcao3" src="media/opcao3.gif" descriptor="dOpcao3"/>
108: <media id="opcao4" src="media/opcao4.gif" descriptor="dOpcao4"/>
109: <media id="opcao5" src="media/opcao5.gif" descriptor="dOpcao5"/>
110: <media id="opcao6" src="media/opcao6.gif" descriptor="dOpcao6"/>
111:
112: <!-----+
113: ! SWITCH:
114: ! contém os nós dentre os quais um será selecionado
115: ! com base nas regras definidas por bindRule
116: !+-----+
117:
118: <switch id="switchOpcao">
119:     <!-- caso a regra ri seja válida, dispara o nó selecao -->
120:
121:     <bindRule rule="r1" constituent="selecao1" />
122:     <bindRule rule="r2" constituent="selecao2" />
123:     <bindRule rule="r3" constituent="selecao3" />
124:     <bindRule rule="r4" constituent="selecao4" />
125:     <bindRule rule="r5" constituent="selecao5" />
126:     <bindRule rule="r6" constituent="selecao6" />

```

```

127:
128: <media id="selecao1" src="media/selecao1.html" descriptor="dResultado"/>
129: <media id="selecao2" src="media/selecao2.html" descriptor="dResultado"/>
130: <media id="selecao3" src="media/selecao3.html" descriptor="dResultado"/>
131: <media id="selecao4" src="media/selecao4.html" descriptor="dResultado"/>
132: <media id="selecao5" src="media/selecao5.html" descriptor="dResultado"/>
133: <media id="selecao6" src="media/selecao6.html" descriptor="dResultado"/>
134:
135: </switch>
136:
137: <!--+-----+
138: ! ELOS
139: ! define os elos que regem o sincronismo entre as mídias
140: !+-----+
141:
142: <!-- início do vídeo 1 deve iniciar a exibição das opções -->
143:
144: <link id="Video1Start" xconnector="connBase#onBeginStartN">
145:   <bind component="video1" role="onBegin"/>
146:   <bind component="opcao1" role="start"/>
147:   <bind component="opcao2" role="start"/>
148:   <bind component="opcao3" role="start"/>
149:   <bind component="opcao4" role="start"/>
150:   <bind component="opcao5" role="start"/>
151:   <bind component="opcao6" role="start"/>
152: </link>
153:
154: <link id="Selecao_OK1" xconnector="onSelectionSetNStartNStopN">
155:   <bind component="opcao1" role="onSelection"/>
156:   <bind component="nodeSettings" interface="opcao" role="set">
157:     <bindParam name="var" value="1"/>
158:   </bind>
159:   <bind component="switchOpcao" role="start"/>
160:   <bind component="video1" role="stop"/>
161:   <bind component="opcao1" role="stop"/>
162:   <bind component="opcao2" role="stop"/>
163:   <bind component="opcao3" role="stop"/>
164:   <bind component="opcao4" role="stop"/>
165:   <bind component="opcao5" role="stop"/>
166:   <bind component="opcao6" role="stop"/>
167: </link>
168:
169: <link id="Selecao_OK2" xconnector="onSelectionSetNStartNStopN">
170:   <bind component="opcao2" role="onSelection"/>
171:   <bind component="nodeSettings" interface="opcao" role="set">
172:     <bindParam name="var" value="2"/>
173:   </bind>
174:   <bind component="switchOpcao" role="start"/>
175:   <bind component="video1" role="stop"/>
176:   <bind component="opcao1" role="stop"/>
177:   <bind component="opcao2" role="stop"/>
178:   <bind component="opcao3" role="stop"/>
179:   <bind component="opcao4" role="stop"/>
180:   <bind component="opcao5" role="stop"/>
181:   <bind component="opcao6" role="stop"/>
182: </link>
183:
184: <link id="Selecao_OK3" xconnector="onSelectionSetNStartNStopN">
185:   <bind component="opcao3" role="onSelection"/>
186:   <bind component="nodeSettings" interface="opcao" role="set">
```

```
187:      <bindParam name="var" value="3"/>
188:    </bind>
189:    <bind component="switchOpcão" role="start"/>
190:    <bind component="video1" role="stop"/>
191:    <bind component="opção1" role="stop"/>
192:    <bind component="opção2" role="stop"/>
193:    <bind component="opção3" role="stop"/>
194:    <bind component="opção4" role="stop"/>
195:    <bind component="opção5" role="stop"/>
196:    <bind component="opção6" role="stop"/>
197:  </link>
198:
199:  <link id="Seleção_OK4" xconnector="onSelectionSetNStartNStopN">
200:    <bind component="opção4" role="onSelection"/>
201:    <bind component="nodeSettings" interface="opção" role="set">
202:      <bindParam name="var" value="4"/>
203:    </bind>
204:    <bind component="switchOpcão" role="start"/>
205:    <bind component="video1" role="stop"/>
206:    <bind component="opção1" role="stop"/>
207:    <bind component="opção2" role="stop"/>
208:    <bind component="opção3" role="stop"/>
209:    <bind component="opção4" role="stop"/>
210:    <bind component="opção5" role="stop"/>
211:    <bind component="opção6" role="stop"/>
212:  </link>
213:
214:  <link id="Seleção_OK5" xconnector="onSelectionSetNStartNStopN">
215:    <bind component="opção5" role="onSelection"/>
216:    <bind component="nodeSettings" interface="opção" role="set">
217:      <bindParam name="var" value="5"/>
218:    </bind>
219:    <bind component="switchOpcão" role="start"/>
220:    <bind component="video1" role="stop"/>
221:    <bind component="opção1" role="stop"/>
222:    <bind component="opção2" role="stop"/>
223:    <bind component="opção3" role="stop"/>
224:    <bind component="opção4" role="stop"/>
225:    <bind component="opção5" role="stop"/>
226:    <bind component="opção6" role="stop"/>
227:  </link>
228:
229:  <link id="Seleção_OK6" xconnector="onSelectionSetNStartNStopN">
230:    <bind component="opção6" role="onSelection"/>
231:    <bind component="nodeSettings" interface="opção" role="set">
232:      <bindParam name="var" value="6"/>
233:    </bind>
234:    <bind component="switchOpcão" role="start"/>
235:    <bind component="video1" role="stop"/>
236:    <bind component="opção1" role="stop"/>
237:    <bind component="opção2" role="stop"/>
238:    <bind component="opção3" role="stop"/>
239:    <bind component="opção4" role="stop"/>
240:    <bind component="opção5" role="stop"/>
241:    <bind component="opção6" role="stop"/>
242:  </link>
243:
244:  </body>
245:
246: </ncl>
```

Exemplo 13 – Reutilização de nós

O exemplo a seguir ilustra a reutilização de um nó de vídeo (nó *videoPrincipal*, linha 133) em dois contextos distintos: como nó de vídeo *videoBasico*, no contexto *ctxBasico* (linha 145), e como nó de vídeo *videoAvancado*, no contexto *ctxAvancado* (linha 223). Em cada contexto são associadas diferentes âncoras para fazer sincronismo com certas mídias.

Listagem 13. Documento NCL ilustrando re-uso num programa com dois modos de exibição — básico e avançado.

```

1:  <?xml version="1.0" encoding="ISO-8859-1"?>
2:
3:  <!--+++++-----+
4:      EXEMPLO 13
5:
6:      Objetivo: Reproduzir um vídeo e permitir que o usuário selecione
7:          o nível de conhecimento sobre um tópico,
8:          através do pressionamento das teclas
9:          vermelha e verde do controle remoto
10:         e com feedback imediato para o usuário.
11:         O vídeo "principal" será re-usado em 2 contextos:
12:             - contexto básico, entrelaçado com vídeos e texto explicativo em diversos pontos
13:             - contexto avançado, somente com texto explicativo em alguns pontos (menos do que no básico)
14:
15:     Características:
16:
17:         - sincronismo: simples (de início e término de mídias)
18:         - interação do usuário: seleção do contexto a ser exibido
19:         - re-uso de nós de conteúdo
20:
21:     Preparação:
22:
23:     Para executar este exemplo, é necessário ter as seguintes mídias no subdiretório
24:     mídia a partir do caminho do arquivo NCL:
25:
26:     1) arquivo de vídeo chamado video1.mpg
27:     2) arquivo de vídeo chamado video_principal.mpg
28:     3) arquivos de vídeo chamados video_basico1.mpg, video_basico2.mpg, video_basico3.mpg
29:     4) arquivos de texto basico1.html, basico2.html, basico3.html
30:     5) arquivos de texto avancado1.html, avancado2.html, avancado3.html
31:     6) arquivo de imagem chamado botao_vermelho.gif, indicando a ação do botão vermelho
32:     7) arquivo de imagem chamado botao_verde.gif, indicando a ação do botão verde
33:
34:     !+++++-----+
35:
36:  <!--+++++-----+
37:  ! CABEÇALHO NCL:
38:  ! define as URIs dos esquemas da NCL
39:  !+++++-----+
40:
41:  <ncl id="exemplo13" xmlns="http://www.ncl.org.br/NCL3.0/EDTVProfile"
42:      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
43:      xsi:schemaLocation="http://www.ncl.org.br/NCL3.0/EDTVProfile
44:                          http://www.ncl.org.br/NCL3.0/profiles/NCL30EDTV.xsd">
45:  !+++++-----+

```

```

46:
47: <head>
48:
49: <!--+-----+
50: ! BASE DE REGIÕES:
51: ! define as regiões na tela onde as mídias são apresentadas
52: !-----+
53:
54: <regionBase>
55: <region id="rgTV" height="576" width="1024">
56:   <region id="rgVideo1" height="480" width="640" top="0" left="0"/>
57:   <region id="rgVideo2" left="384" bottom="0" width="640" height="480" />
58:   <region id="rgTexto1" height="50" width="384" top="480" left="0"/>
59:   <region id="rgBotaoVermelho" left="200" top="420" width="25" height="25" />
60:   <region id="rgBotaoVerde" left="200" top="470" width="25" height="25" />
61: </region>
62: </regionBase>
63:
64: <!--+-----+
65: ! BASE DE DESCRIPTORES:
66: ! define como as mídias são apresentadas
67: !-----+
68:
69: <descriptorBase>
70:   <descriptor id="dVideo1" region="rgVideo1" />
71:   <descriptor id="dVideo2" region="rgVideo2" />
72:
73:   <descriptor id="dTexto1" region="rgTexto1" />
74:
75:   <descriptor id="dBotaoVermelho" region="rgBotaoVermelho" />
76:   <descriptor id="dBotaoVerde" region="rgBotaoVerde" />
77: </descriptorBase>
78:
79: <!--+-----+
80: ! BASE DE CONECTORES:
81: ! definem o comportamento dos elos
82: !-----+
83:
84: <connectorBase>
85:   <importBase alias="connBase" documentURI="maestroConnectorBase.conn"/>
86:
87:
88:   <causalConnector id="onKeySelectionStartNStopNAbortN">
89:     <connectorParam name="keyCode" />
90:     <connectorParam name="delay" />
91:
92:     <simpleCondition role="onSelection" key="$keyCode"/>
93:
94:     <compoundAction operator="seq">
95:
96:       <simpleAction role="dstart" eventType="presentation" actionType="start" delay="$delay"
97:         max="unbounded" qualifier="par"/>
98:         <simpleAction role="dstop" eventType="presentation" actionType="stop" delay="$delay"
99:           max="unbounded" qualifier="par"/>
100:          <simpleAction role="dabort" eventType="presentation" actionType="abort" delay="$delay"
101:            max="unbounded" qualifier="par"/>
102:            <simpleAction role="start" delay="$delay" max="unbounded" qualifier="par"/>
103:            <simpleAction role="stop" delay="$delay" max="unbounded" qualifier="par"/>

```

```

103: <simpleAction role="abort" delay="$delay" max="unbounded" qualifier="par"/>
104: -->
105:
106: </compoundAction>
107: </causalConnector>
108:
109: </connectorBase>
110:
111: </head>
112:
113: <!-----+
114: ! CORPO DO PROGRAMA:
115: ! define as mídias e estrutura do programa
116: !+++++
117:
118: <body>
119:
120: <!-----+
121: ! PONTO DE ENTRADA:
122: ! indica o componente onde o programa inicia
123: !+++++
124:
125: <port id="pInicio" component="video1" />
126:
127: <!-----+
128: ! MÍDIAS:
129: ! define o local dos arquivos de mídia e as associa com seus descritores
130: !+++++
131: <media type="video" id="video1" src="media/video1.mpg" descriptor="dVideo1" />
132:
133: <media type="video" id="videoPrincipal" src="media/video_principal.mpg" descriptor="dVideo1" />
134:
135: <media type="image" id="botaoVermelho" src="media/botao_vermelho.gif" descriptor="dBotaoVermelho" />
136: <media type="image" id="botaoVerde" src="media/botao_verde.gif" descriptor="dBotaoVerde" />
137:
138: <!-----+
139: ! CONTEXTOS:
140: ! os documentos básico e avançado
141: !+++++
142: <context id="ctxBasico">
143:   <port id="pBasico" component="videoBasico" />
144:
145:   <media id="videoBasico" refer="videoPrincipal">
146:     <!-- âncoras no vídeo que devem ser sincronizadas com a legenda -->
147:     <area id="aVideoBasicoTexto1" begin="1s" end="3s" />
148:     <area id="aVideoBasicoTexto2" begin="5s" end="15s" />
149:     <area id="aVideoBasicoTexto3" begin="16s" end="21s" />
150:     <area id="aVideoBasicoVideo1" begin="2s" end="4s" />
151:     <area id="aVideoBasicoVideo2" begin="8s" end="10s" />
152:     <area id="aVideoBasicoVideo3" begin="13s" end="16s" />
153:   </media>
154:   <media id="textoBasico1" type="text" src="media/basico1.html" descriptor="dTexto1" />
155:   <media id="textoBasico2" type="text" src="media/basico2.html" descriptor="dTexto1" />
156:   <media id="textoBasico3" type="text" src="media/basico3.html" descriptor="dTexto1" />
157:
158:   <media id="videoBasico1" type="video" src="media/video_basico1.mpg" descriptor="dVideo2" />
159:   <media id="videoBasico2" type="video" src="media/video_basico2.mpg" descriptor="dVideo2" />
160:   <media id="videoBasico3" type="video" src="media/video_basico3.mpg" descriptor="dVideo2" />
161:
162:   <!-- links para sincronizar os textos com o vídeo -->

```

```

163: <link id="IBasicoTexto1Start" xconnector="connBase#onBeginStartN">
164:   <bind component="videoBasico" interface="aVideoBasicoTexto1" role="onBegin" />
165:   <bind component="textoBasico1" role="start" />
166: </link>
167: <link id="IBasicoTexto2Start" xconnector="connBase#onBeginStartN">
168:   <bind component="videoBasico" interface="aVideoBasicoTexto2" role="onBegin" />
169:   <bind component="textoBasico2" role="start" />
170: </link>
171: <link id="IBasicoTexto3Start" xconnector="connBase#onBeginStartN">
172:   <bind component="videoBasico" interface="aVideoBasicoTexto3" role="onBegin" />
173:   <bind component="textoBasico3" role="start" />
174: </link>
175:
176: <link id="IBasicoTexto1End" xconnector="connBase#onEndStopN">
177:   <bind component="videoBasico" interface="aVideoBasicoTexto1" role="onEnd" />
178:   <bind component="textoBasico1" role="stop" />
179: </link>
180: <link id="IBasicoTexto2End" xconnector="connBase#onEndStopN">
181:   <bind component="videoBasico" interface="aVideoBasicoTexto2" role="onEnd" />
182:   <bind component="textoBasico2" role="stop" />
183: </link>
184: <link id="IBasicoTexto3End" xconnector="connBase#onEndStopN">
185:   <bind component="videoBasico" interface="aVideoBasicoTexto3" role="onEnd" />
186:   <bind component="textoBasico3" role="stop" />
187: </link>
188:
189: <!-- links para sincronizar os vídeos auxiliares com o vídeo principal -->
190: <link id="IBasicoVideo1Start" xconnector="connBase#onBeginStartNPauseN">
191:   <bind component="videoBasico" interface="aVideoBasicoVideo1" role="onBegin" />
192:   <bind component="videoBasico" role="pause" />
193:   <bind component="videoBasico1" role="start" />
194: </link>
195: <link id="IBasicoVideo2Start" xconnector="connBase#onBeginStartNPauseN">
196:   <bind component="videoBasico" interface="aVideoBasicoVideo2" role="onBegin" />
197:   <bind component="videoBasico" role="pause" />
198:   <bind component="videoBasico2" role="start" />
199: </link>
200: <link id="IBasicoVideo3Start" xconnector="connBase#onBeginStartNPauseN">
201:   <bind component="videoBasico" interface="aVideoBasicoVideo3" role="onBegin" />
202:   <bind component="videoBasico" role="pause" />
203:   <bind component="videoBasico3" role="start" />
204: </link>
205:
206: <link id="IBasicoVideo1End" xconnector="connBase#onEndResumeN">
207:   <bind component="videoBasico1" role="onEnd" />
208:   <bind component="videoBasico" role="resume" />
209: </link>
210: <link id="IBasicoVideo2End" xconnector="connBase#onEndResumeN">
211:   <bind component="videoBasico2" role="onEnd" />
212:   <bind component="videoBasico" role="resume" />
213: </link>
214: <link id="IBasicoVideo3End" xconnector="connBase#onEndResumeN">
215:   <bind component="videoBasico3" role="onEnd" />
216:   <bind component="videoBasico" role="resume" />
217: </link>
218: </context>
219:
220: <context id="ctxAvancado">
221:   <port id="pAvancado" component="videoAvancado" />
222:

```

```

223: <media id="videoAvancado" refer="videoPrincipal">
224:   <!-- âncoras no vídeo que devem ser sincronizadas com a legenda -->
225:   <area id="aVideoAvancadoTexto1" begin="5s" end="9s"/>
226:   <area id="aVideoAvancadoTexto2" begin="10s" end="14s"/>
227:   <area id="aVideoAvancadoTexto3" begin="15s" end="19s"/>
228: </media>
229: <media id="textoAvancado1" type="text" src="media/avancado1.html" descriptor="dTexto1" />
230: <media id="textoAvancado2" type="text" src="media/avancado2.html" descriptor="dTexto1" />
231: <media id="textoAvancado3" type="text" src="media/avancado3.html" descriptor="dTexto1" />
232:
233: <!-- links para sincronizar os textos com o vídeo -->
234: <link id="IAvancadoTexto1Start" xconnector="connBase#onBeginStartN">
235:   <bind component="videoAvancado" interface="aVideoAvancadoTexto1" role="onBegin" />
236:   <bind component="textoAvancado1" role="start" />
237: </link>
238: <link id="IAvancadoTexto2Start" xconnector="connBase#onBeginStartN">
239:   <bind component="videoAvancado" interface="aVideoAvancadoTexto2" role="onBegin" />
240:   <bind component="textoAvancado2" role="start" />
241: </link>
242: <link id="IAvancadoTexto3Start" xconnector="connBase#onBeginStartN">
243:   <bind component="videoAvancado" interface="aVideoAvancadoTexto3" role="onBegin" />
244:   <bind component="textoAvancado3" role="start" />
245: </link>
246:
247: <link id="IAvancadoTexto1End" xconnector="connBase#onEndStopN">
248:   <bind component="videoAvancado" interface="aVideoAvancadoTexto1" role="onEnd" />
249:   <bind component="textoAvancado1" role="stop" />
250: </link>
251: <link id="IAvancadoTexto2End" xconnector="connBase#onEndStopN">
252:   <bind component="videoAvancado" interface="aVideoAvancadoTexto2" role="onEnd" />
253:   <bind component="textoAvancado2" role="stop" />
254: </link>
255: <link id="IAvancadoTexto3End" xconnector="connBase#onEndStopN">
256:   <bind component="videoAvancado" interface="aVideoAvancadoTexto3" role="onEnd" />
257:   <bind component="textoAvancado3" role="stop" />
258: </link>
259: </context>
260:
261:
262: <!--+-----+
263: ! ELOS
264: ! define os elos que regem o sincronismo entre as mídias
265: !+-----+
266:
267: <!-- início do video1 deve exibir botões -->
268: <link id="IVideo1Init" xconnector="connBase#onBeginStartN">
269:   <bind component="video1" role="onBegin" />
270:   <bind component="botaoVermelho" role="start" />
271:   <bind component="botaoVerde" role="start" />
272: </link>
273:
274: <!-- término do video1 deve dispará-lo novamente (deve tocar em loop) -->
275: <link id="IVideo1Loop" xconnector="connBase#onEndStartN">
276:   <bind component="video1" role="onEnd" />
277:   <bind component="video1" role="start" />
278: </link>
279:
280: <!-- define nível básico quando a tecla vermelha é pressionada -->
281:
282: <!--

```

```

283: <link id="lSelectBotaoVermelho" xconnector="onKeySelectionStartNStopNAbrtN">
284:   <linkParam name="delay" value="0.5s"/>
285:   <bind component="botaoVermelho" role="onSelection">
286:     <bindParam name="keyCode" value="RED" />
287:   </bind>
288:   <bind component="botaoVerde" role="stop" />
289:   <bind component="botaoVermelho" role="stop">
290:     <bindParam name="delay" value="5s" />
291:   </bind>
292:   <bind component="video1" role="abort" />
293:   <bind component="ctxBasico" interface="pBasico" role="start">
294:     <bindParam name="delay" value="5s" />
295:   </bind>
296: </link>
297: -->
298:   <!-- define nível avançado quando a tecla verde é pressionada -->
299:
300: <!--
301:   <link id="lSelectBotaoVerde" xconnector="onKeySelectionStartNStopNAbrtN">
302:     <linkParam name="delay" value="0.5s" />
303:     <bind component="botaoVerde" role="onSelection">
304:       <bindParam name="keyCode" value="GREEN" />
305:     </bind>
306:     <bind component="botaoVerde" role="stop">
307:       <bindParam name="delay" value="5s" />
308:     </bind>
309:     <bind component="botaoVermelho" role="stop" />
310:     <bind component="video1" role="abort" />
311:     <bind component="ctxAvancado" interface="pAvancado" role="start">
312:       <bindParam name="delay" value="5s" />
313:     </bind>
314:   </link>
315: -->
316:
317:   <!-- define nível básico quando a tecla vermelha é pressionada -->
318:
319: <link id="lSelectBotaoVermelho" xconnector="onKeySelectionStartNStopNAbrtN">
320:   <linkParam name="delay" value="0.5s" />
321:   <bind component="botaoVermelho" role="onSelection">
322:     <bindParam name="keyCode" value="RED" />
323:   </bind>
324:   <bind component="botaoVerde" role="dstop" />
325:   <bind component="botaoVermelho" role="dstop">
326:     <bindParam name="delay" value="5s" />
327:   </bind>
328:   <bind component="video1" role="dabort" />
329:   <bind component="ctxBasico" interface="pBasico" role="dstart">
330:     <bindParam name="delay" value="5s" />
331:   </bind>
332: </link>
333:
334:
335:   <!-- define nível avançado quando a tecla verde é pressionada -->
336:
337:
338: <link id="lSelectBotaoVerde" xconnector="onKeySelectionStartNStopNAbrtN">
339:   <linkParam name="delay" value="0.5s" />
340:   <bind component="botaoVerde" role="onSelection">
341:     <bindParam name="keyCode" value="GREEN" />
342:   </bind>

```

```
343: <bind component="botaoVerde" role="dstop">
344:   <bindParam name="delay" value="5s"/>
345: </bind>
346: <bind component="botaoVermelho" role="dstop" />
347: <bind component="video1" role="dabort" />
348: <bind component="ctxAvancado" interface="pAvancado" role="dstart">
349:   <bindParam name="delay" value="5s"/>
350: </bind>
351: </link>
352:
353: </body>
354: </ncl>
```


Apêndice II – Conectores Predefinidos

O arquivo maestroConnectorBase.conn, distribuído com a ferramenta Composer, define diversos conectores de uso freqüente. Para a construção de boa parte dos programas audiovisuais interativos, não é necessário criar conectores, mas sim conhecer os que já vêm definidos. Este apêndice apresenta os conectores do arquivo maestroConnectorBase.conn.

Os conectores mais simples envolvem apenas uma condição e uma ação. Em geral, podem ser lidos da seguinte maneira: Quando <condição> ocorrer, realize <ação>. Por convenção, os identificadores do conector refletem a condição e a ação. Dessa maneira, um conector simples é definido de acordo com o seguinte modelo:

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 da mídia associada ao papel **start**.”

Os conectores predefinidos que seguem esse modelo podem ser vistos na seguinte tabela:

condição	ação	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	

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:

Conecor com ação de atribuição

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

Exemplo de elo que utiliza o conector

```
<link id="IBeginVideo"
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.”

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

Conecotor com condição de acionamento de tecla

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

Exemplo de elo que utiliza o conector

```
<link id="lRedStart"
xconnector="connBase#onKeySelectionStart">
  <bind component="video1" role="onSelection">
    <bindParam name="keyCode" value="RED"/>
  </bind>
  <bind component="video2" role="start" />
</link>
```

"Quando a tecla vermelha for pressionada durante a apresentação de video1, inicie a exibição de video2."

Até agora, vimos conectores em que cada papel pode ser mapeado a apenas uma mídia. Mas existem conectores predefinidos que permitem múltiplos mapeamentos para um único papel. Esses conectores seguem o seguinte modelo (observe que o **id** do conector de exemplo mudou para incorporar um N após o nome da ação que possui cardinalidade N):

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 número 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** associadas 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="lBeginVideo1" xconnector="onBeginStartN">
  <bind component="video1" role="onBegin" />

  <bind component="imagem1" role="start" />
  <bind component="imagem2" role="start" />
</link>
```

Até agora, vimos condições e ações simples. No entanto, por diversas vezes precisamos realizar ações de diferentes tipos ao mesmo tempo. Para isto, definimos conectores com ações compostas, como no seguinte modelo:

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" />
  </compoundAction>
</causalConnector>
```

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

Um recurso interessante é a possibilidade de se introduzir retardos na realização de uma ação. No conector a seguir, por exemplo, é possível definir em quanto tempo após a ativação do elo as mídias associadas ao papel **stop** vão ter sua apresentação terminada:

```
<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>
```

Além de ações compostas, 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 freqüência, definem-se conectores que testam o valor de uma determinada propriedade para decidir sobre a ativação do elo. No conector a seguir, por exemplo, 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"/>
  <compoundCondition operator="and">
    <simpleCondition role="onBegin" />
    <assessmentStatement comparator="eq">
      <attributeAssessment role="attNodeTest" eventType="attribution" attributeType="nodeAttribute"/>
      <valueAssessment value="$value"/>
    </assessmentStatement>
  </compoundCondition>
  <simpleAction role="start" max="unbounded" qualifier="par"/>
</causalConnector>
```

A lista completa dos conectores predefinidos no arquivo maestroConnector.conn é a seguinte:

onBeginAttNodeTestSetNStartN	onKeySelectionNAttNodeTestStartN
onBeginAttNodeTestStartN	onKeySelectionNAttNodeTestStopN
onBeginAttNodeTestStartNStopN	onKeySelectionNPauseN
onBeginAttNodeTestStopN	onKeySelectionNResumeN
onBeginPause	onKeySelectionNSet
onBeginPauseN	onKeySelectionNStartN
onBeginResume	onKeySelectionNStartNStopN
onBeginResumeN	onKeySelectionNStopN
onBeginSet	onKeySelectionNStopNStartN
onBeginSetN	onKeySelectionPause
onBeginSetVarNDelayStopN	onKeySelectionPauseN
onBeginStart	onKeySelectionResume
onBeginStartN	onKeySelectionResumeN
onBeginStartNDelayStopN	onKeySelectionSet
onBeginStartNPauseN	onKeySelectionSetN
onBeginStartNResumeN	onKeySelectionSetVarNDelayStopN
onBeginStartNSetN	onKeySelectionStart
onBeginStartNStopN	onKeySelectionStartN
onBeginStop	onKeySelectionStartNDelayStopN
onBeginStopN	onKeySelectionStartNPauseN
onBeginStopNDelayStartN	onKeySelectionStartNResumeN
onBeginStopNPauseN	onKeySelectionStartNSetN
onBeginStopNResumeN	onKeySelectionStartNStopN
onBeginStopNSetN	onKeySelectionStop
onBeginStopNStartN	onKeySelectionStopN
onEndAttNodeTestSetNStopN	onKeySelectionStopNDelayStartN
onEndAttNodeTestStartN	onKeySelectionStopNPauseN
onEndAttNodeTestStartNStopN	onKeySelectionStopNResumeN
onEndAttNodeTestStopN	onKeySelectionStopNSetN
onEndPause	onKeySelectionStopNStartN
onEndPauseN	onSelectionPause
onEndResume	onSelectionPauseN
onEndResumeN	onSelectionResume
onEndSet	onSelectionResumeN
onEndSetN	onSelectionSet
onEndSetVarNDelayStopN	onSelectionSetN
onEndStart	onSelectionSetVarNDelayStopN
onEndStartN	onSelectionStart
onEndStartNDelayStopN	onSelectionStartN
onEndStartNPauseN	onSelectionStartNDelayStopN
onEndStartNResumeN	onSelectionStartNPauseN
onEndStartNSetN	onSelectionStartNResumeN
onEndStartNStopN	onSelectionStartNSetN
onEndStop	onSelectionStartNStopN
onEndStopN	onSelectionStop
onEndStopNDelayStartN	onSelectionStopN
onEndStopNPauseN	onSelectionStopNDelayStartN
onEndStopNResumeN	onSelectionStopNPauseN
onEndStopNSetN	onSelectionStopNResumeN
onEndStopNStartN	onSelectionStopNSetN
	onSelectionStopNStartN