Estendendo NCL: objetos NCLua como exibidores para novos tipos de mídia

José Geraldo de Sousa Junior[1] Roberto Gerson de Albuquerque Azevedo[1] Carlos de Salles Soares Neto[1,2] Luiz Fernando Gomes Soares[1]

[1] Laboratório Telemídia – Departamento de Informática – PUC-Rio
Rua Marquês de São Vicente 225
22453-900 Rio de Janeiro, RJ
+55-21-3527-1500 Ext: 3503

[2] Laboratory of Advanced Web Systems – Departamento de Informática – UFMA
Av. Dos Portugueses s/n – Campus do Bacanga
65085-000 São Luis, MA
+55-98-3301-8223

{josegeraldo, robertogerson}@telemidia.puc-rio.br, csalles@deinf.ufma.br, lfgs@inf.puc-rio.br

ABSTRACT

The Brazilian Digital Terrestrial TV System (SBTVD-T) defines support for several media types of codification (audio, video, images and so on), which can be presented in an interactive application using its middleware Ginga. In order to establish relationships and to synchronize these media objects, Ginga-NCL, which is the Ginga declarative subsystem, uses Nested Context Language (NCL). This article aims to present NCL as an extensible language, since that it allows new media types, those which have not been standardized by SBTVD-T, to be played. NCL is extended by using Lua imperative objects implementing players for those new media types. As an use case, a SubRip Subtitle Format (SRT) player is presented. SubRip is a codification not included by SBTVD-T standards. The work's secondary goal is to reaffirm NCL characteristics as a glue language, showing that NCL separates media contents from its relationship specifications.

RESUMO

O Sistema Brasileiro de TV Digital Terrestre (SBTVD-T) determina quais são as diversas codificações para tipos de mídia (áudio, vídeo, imagens etc.) que podem estar presentes em uma aplicação interativa suportada por seu *middleware* Ginga. Para relacionar e sincronizar esses objetos de mídia, o Ginga-NCL, subsistema declarativo do Ginga, utiliza a linguagem NCL (*Nested Context Language*). A proposta deste artigo é apresentar NCL como uma linguagem extensível, ao possibilitar a exibição de novos tipos de mídia que não foram padronizados pelo SBTVD-T. A extensão é realizada por meio de objetos imperativos Lua (objetos NCLua) que implementam exibidores para esses novos tipos de mídia. Como caso de uso, é apresentado um exibidor de legendas no formato SubRip (*SRT)*, codificação não reconhecida pelo sistema. Um objetivo secundário é reforçar o caráter de NCL como uma linguagem de cola, demonstrando que é possível relacionar mídias independentemente de seus tipos específicos.

Webmedia 2010, Outubro 5–8, 2010, Belo Horizonte, Minas Gerais, Brasil.

Categories and Subject Descriptors

D.3.2 [**Programming Languages**]: Language Classifications – *Extensible languages, Multiparadigm languages, Specialized application languages.*

General Terms

Design, Languages, Standardization.

Keywords

NCL, Lua, Digital TV, Ginga, middlewares, declarative languages, multimedia systems, players, extensibility.

1. INTRODUÇÃO

Nested Context Language (NCL) é uma linguagem declarativa de domínio específico – denominada "linguagem declarativa" no decorrer do texto – que permite a especificação de documentos multimídia compostos por objetos de mídia sincronizados espaço-temporalmente entre si. NCL atua como uma linguagem de cola, promovendo um ambiente em que fica clara a separação entre o conteúdo das mídias e a especificação dos relacionamentos de sincronismo entre elas. [1]

O *middleware* Ginga [2], padrão do Sistema Brasileiro de TV Digital (SBTVD-T), é constituído por dois subsistemas principais: Ginga-J, responsável por executar aplicações imperativas, em Java; e Ginga-NCL, responsável por orquestrar aplicações declarativas, em NCL. Na arquitetura do Ginga-NCL [3], a separação entre conteúdo de mídia e relacionamentos é obtida principalmente por meio dos *exibidores* (ou *players de mídia*). Os exibidores possuem funções específicas associadas ao tipo de mídia a que se referem. Há, por exemplo, diferentes exibidores, um para cada tipo de objeto de mídia, tais como imagem, vídeo, audio etc. O *formatador* (ou orquestrador) é o responsável por controlar o sincronismo temporal e espacial da apresentação NCL. Para isso, o formatador deve controlar esses diversos exibidores, por exemplo, carregando-os quando necessário, informando-os que devem iniciar a exibição de uma mídia (e qual é essa mídia), quando deve pará-la etc.

Um exibidor, em especial, é o responsável pelo tratamento de objetos imperativos codificados na linguagem de *script* Lua [4], os denominados NCLua. Utilizar uma linguagem de *script* imperativa associada a linguagens declarativas é bastante útil, já que amplia a expressividade dessas últimas [4]. Objetos NCLua se comunicam com documentos NCL por meio do paradigma

orientado a eventos. O modelo de execução NCLua consiste em tratar eventos que são enviados do formatador para objetos NCLua e, da mesma forma, esses objetos também podem notificar eventos ao formatador.

Em objetos de mídia não-imperativos (áudio, vídeo, imagem etc) a lógica das ações a serem executadas quando cada evento é enviado pelo formatador ao exibidor já é pré-determinada pelo tipo daquele objeto de mídia. Em objetos de mídia imperativos, por outro lado, o autor da mídia (no caso, a mídia é o código Lua) é o responsável por tratar os diversos eventos enviados pelo formatador, tais como *start*, *stop*, *set* etc, e determinar a lógica que melhor lhe convier para executar ações baseadas na ocorrência desses eventos. Para isso, o autor do objeto NCLua faz uso do módulo *event*, assunto abordado na Seção 3. Como o autor do objeto NCLua é quem determina a lógica do tratamento dos eventos, abre-se também a possibilidade de ter esses objetos NCLua como exibidores para novos tipos de mídia.

A proposta deste artigo é ilustrar esse importante mecanismo de extensão de NCL, que tira proveito do uso de objetos NCLua. O objetivo é demonstrar como autores podem criar exibidores para novos tipos de mídia que não foram previstos na implementação de referência do Ginga-NCL[1] ou na norma ABNT NBR 15606-1 [2] do SBTVD. A grande vantagem dessa proposta é que, com ela, as aplicações NCL podem se beneficiar de determinados tipos de mídia, e, ainda assim, manter a compatibilidade com a norma do SBTVD-T. O artigo apresenta, como caso de uso desse mecanismo de extensão, a implementação de um exibidor para arquivos *SubRip Subtitle Format* (*SRT)* [5] em NCLua. O exibidor é codificado segundo um padrão de *interface* entre documentos NCL e exibidores em NCLua apresentado na Seção 3.1. Esse padrão é uma contribuição deste trabalho como um mecanismo de reúso que torna mais rápida a implementação de exibidores. O reúso é explorado na implementação de um exibidor de GIF animado[6].

As seções seguintes têm o objetivo de dar sustentação ao que foi mencionado sobre a extensibilidade de NCL e apresentar o caso de uso mencionado no parágrafo anterior. Na Seção 2, é apresentado o modelo de execução de objetos NCLua. A Seção 3 situa o objeto NCLua com função de exibidor na arquitetura do Ginga-NCL, e mostra como se dá a comunicação entre o documento NCL e esse exibidor. A Seção 4 apresenta um caso de uso, que é um exibidor de legendas de arquivos *SubRip*. Finalmente, a Seção 5 apresenta as considerações finais e trabalhos futuros.

2. MODELO DE COMUNICAÇÃO ENTRE NCL E EXIBIDORES DE MÍDIAS

Os relacionamentos entre mídias em um documento NCL são especificados por meio de entidades denominadas *elos*. Os elos definem relacionamentos entre eventos. Tomando como exemplo, é possível especificar um elo que descreve a seguinte sentença: quando uma tecla for pressionada (evento de seleção), inicie um objeto de imagem (evento de apresentação). NCL define algumas classes básicas de evento, tais como evento de exibição, de seleção e de atribuição. O evento, na dimensão do tempo, pode assumir um estado corrente, tal como "preparado", "pausado" ou "ocorrendo". Ações de inicio, fim, pausa e recomeço são possíveis sobre um evento, o que resulta na transição de um estado para outro.

A alteração no estado corrente de um evento, ocasionada pelo disparo de um elo, deve ser comunicada ao exibidor responsável pela apresentação deste tipo de mídia. Adicionalmente, os exibidores também devem notificar ao formatador a ocorrência de eventos nas mídias, como, por exemplo, o seu fim natural. No caso de objetos imperativos, é o próprio programador o responsável por implementar esse controle.

O código NCL da Listagem 1 possui um elo que inicia o objeto de imagem *img2* assim que o objeto *img1* começar. O formatador, no momento em que esse elo é disparado, comunica ao exibidor de imagens *png* que ele deve iniciar a exibição da imagem *img2*.

```
1: <media id="img1" src="img1.png"
                    type="image/png" />
2: <media id="img2" src="img2.png"
                    type="image/png"/>
3: ...
4: <link xconnector="onBeginStart">
5:   <bind role="onBegin" component="img1"/>
6:   <bind role="start" component="img2"/>
7: </link>
```

Listagem 1. Elo escrito em NCL

Os exibidores previstos pelo SBTVD-T são capazes de se comunicar com o formatador do Ginga-NCL como mencionado no parágrafo anterior. Alguns deles são exibidores de imagem *png* e *jpeg* e os de vídeo *mpeg.* Contudo, tipos de mídia que não estão previstos no SBTVD-T não podem ser exibidos, não havendo um exibidor disponível para aquela mídia. O código da Listagem 2, por exemplo, não pode ser apresentado em um formatador de referência Ginga-NCL. Isso porque o tipo de arquivo *SubRip* definido para o objeto *srt1* não é suportado, ou seja, não há um exibidor de *SubRip* com o qual o formatador consiga se comunicar. Nota-se que a especificação do sincronismo entre as mídias é feita independente dos seus tipos, o que reforça a característica NCL de isolar o conteúdo das mídias e o documento em si.

```
1:  <media id="img1" src="img1.png"
      type="image/png"/>
2: <media id="srt1" src="subtitle.srt"
     type="text/srt"/>
3: ...
4: <link xconnector="onBeginStart">
5:   <bind role="onBegin" component="img1"/>
6:   <bind role="start" component="srt1"/>
7: </link>
```

Listagem 2. Elo escrito em NCL para um tipo de mídia não suportado pelo Ginga-NCL

2.1 Modelo de execução NCLua

Conforme já mencionado, um dos tipos de mídia previstos no SBTVD-T são *scripts* imperativos especificados em Lua, os objetos NCLua. Para adicionar objetos NCLua em documentos NCL é necessário definir um elemento <media> cujo conteúdo (localizado pelo atributo *src*) é o código Lua a ser executado [2]. Um objeto NCLua pode também definir âncoras de conteúdo (elementos <area>), que representam uma porção do conteúdo da mídia, e âncoras de propriedades (elementos <property>).

A possibilidade de adicionar objetos NCLua em documentos NCL permite estabelecer relacionamentos entre objetos NCLua e objetos de outros tipos de mídia. Na Listagem 2, se o objeto de

[1] Disponível em http://www.softwarepublico.gov.br

mídia com id *srt1* fosse um objeto NCLua, a ação de iniciar realizada pelo elo seria comunicada a este objeto NCLua exatamente igual a qualquer outro tipo de mídia.

Os objetos NCLua se comunicam com entidades externas por meio de eventos (módulo *event* do NCLua). São exemplos de entidades externas: o formatador NCL, emissora, o canal de retorno e os dispositivos de entrada. Os eventos NCLua são representados por tabelas Lua. A Listagem 3 apresenta um exemplo de um evento de apresentação, informando o início do NCLua. Os eventos NCLua são diferenciados por classes de eventos, informando de qual ou para qual entidade eles devem ser enviados. Alguns exemplos de classes de eventos padronizadas pelo Ginga-NCL são: *ncl*, *edit*, *si*, *user*, *key*, *tcp* e *sms*. Neste artigo, há interesse principalmente nos eventos trocados entre o formatador NCL e os objetos NCLua, das classes *ncl*, *edit* e *user*.

```
1:  evt = {
2:       class = 'ncl',
3:       type = 'presentation',
4:       action = 'start'
5:  }
```

Listagem 3. Representação de evento NCLua

Para receber os eventos notificados por alguma entidade externa, o objeto NCLua deve inicialmente registrar (por meio da função *event.register*) pelo menos uma função tratadora de eventos (*callback*). Os vários eventos recebidos de todas as entidades externas são escalonados em uma fila de eventos. A fila FIFO de eventos é esvaziada passando-se cada evento para a primeira função tratadora como parâmetro. Caso essa função tratadora retorne algo diferente de *true*, o evento também é passado para as próximas funções tratadoras. Apenas um evento é tratado por vez.

Eventos NCL (classe *ncl*) são recebidos pelo objeto NCLua quando algum elo NCL dispara uma transição de estado em um evento do objeto NCLua. Da mesma forma, o próprio objeto NCLua também pode informar ao formatador mudanças no seu estado corrente. Para isso, ele deve chamar a função *event.post* passando um evento da classe *ncl* como parâmetro. Adicionalmente, eventos de usuário (classe *user*) permitem aos objetos NCLua estenderem sua funcionalidade criando seus próprios eventos para uso interno. Eventos de comando de edição (classe *edit*) também podem ser enviados ao documento NCL. A Figura 1, a seguir, mostra esquematicamente o processo de comunicação entre o formatador NCL e objetos NCLua.

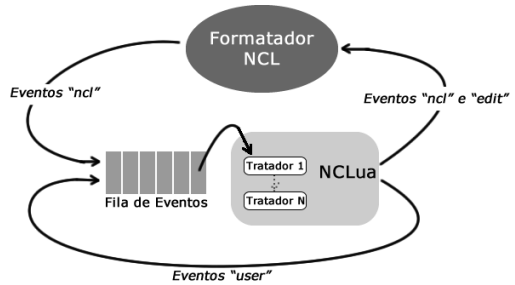

Figura 1. Comunicação NCL-NCLua.

Se um evento de apresentação (*class=ncl, type=presentation*) de uma âncora NCLua for disparado por um elo, as funções registradas para receber notificações de eventos são disparadas no código do objeto NCLua. Essas funções podem tratar os eventos da forma que lhe convier. Adicionalmente, o código NCLua pode também disparar eventos de apresentação em âncoras, que serão notificados ao formatador NCL. O disparo de um evento no NCLua pode ser utilizado como condição em um elo NCL para disparar ações em outros objetos NCL.

Elos também podem disparar eventos de atribuição (*class=ncl, type=attribution*) no objeto NCLua. Tais eventos podem ser mapeados para uma variável do código desse objeto. A ação "*set*", aplicada a uma propriedade do NCLua, causa a atualização do valor da variável relacionada àquela propriedade, juntamente com a chamada das funções tratadoras. Igualmente, quando um evento de seleção (*class=ncl, type=selection*) é identificado pelo formatador NCL, as funções tratadoras são disparadas.

Objetos NCLua estão aptos a fazer operações gráficas na sua área de apresentação (módulo *canvas*), comunicar com entidades externas via tcp (eventos da classe *tcp*) e também fazer a edição ao vivo de documentos NCL (eventos da classe *edit*). Essas facilidades são também exploradas neste artigo.

3. OBJETOS NCLUA COMO EXIBIDORES DE NOVOS TIPOS DE MÍDIA

A seção anterior demonstra que o modelo de execução de objetos NCLua provê um meio de realizar a comunicação entre documentos NCL e esses objetos. Essa comunicação, juntamente com uma API de desenho (por meio do módulo *canvas*) é suficiente para permitir simular exibidores de novos tipos de mídias com objetos NCLua. Dado um tipo de mídia não suportado pelo SBTVD-T, é possível distribuir juntamente com a aplicação NCL, o código de objetos NCLua que reconheçam a codificação dessas mídias e a apresentem na tela.

O autor deve ser apto a implementar esses objetos da melhor forma que lhe convier, recebendo os eventos enviados pelo formatador NCL e resultando nas respectivas ações para aquele tipo de mídia. A Figura 2, por exemplo, mostra o seguinte cenário: um elo dispara a ação de *start* sobre um evento de apresentação de um objeto NCLua, que possui função tratadora de eventos com nome *handler*. A função *handler* analisa que a ação sobre o evento é *start*, e, posteriormente, invoca a parte do código que executa uma função correspondente a iniciar a exibição daquele tipo de mídia. O exemplo da Figura 2 ilustra a função registrada *handler* e a função em destaque, *startNewMediaType*, que possui a lógica de iniciar um tipo de mídia.

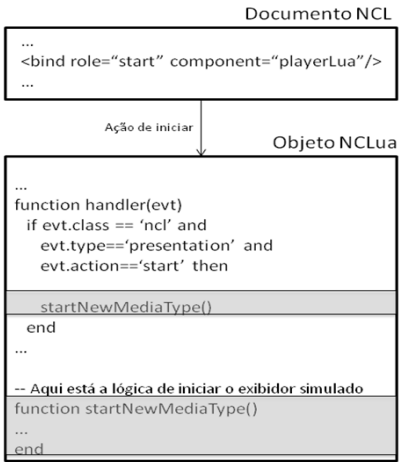

Figura 2. Comunicação entre um documento NCL e um objeto NCLua que simula um exibidor de um tipo de mídia genérico

3.1 Um padrão de interface entre documentos NCL e exibidores em NCLua

Neste trabalho, um padrão de interface entre o código Lua e documentos NCL é sugerido como forma de simplificar a criação de novos exibidores. Esse padrão sugere a definição de funções-padrão para as transições de evento comuns (*start*, *stop*, *pause* etc) atuando sobre o objeto ou sobre âncoras dele. Dessa forma, apenas o conteúdo dessas funções, que são agrupados na tabela Lua *mediaplayer*, precisa ser implementado para cada novo exibidor. Seguindo esse modelo, o autor não necessita, por exemplo, criar as funções tratadoras.

A Figura 3 mostra esquematicamente o objeto NCLua proposto como modelo para implementação de exibidores. A função tratadora é única para todos os exibidores, de todos os tipos de mídia. Ela é responsável por receber os eventos do documento NCL e mapear nas funções correspondentes da tabela *mediaplayer*, que deve ser criada especificamente para cada tipo de mídia. A função tratadora também pode realizar computações adicionais. Por exemplo, ao receber um evento de atribuição em uma propriedade *path*, além de chamar a respectiva função na tabela passando o valor recebido como evento – *mediaplayer.properties.path(value)* –, a função tratadora também é responsável por enviar o evento de final de atribuição para o formatador NCL.

Visando atualizar o conteúdo que está sendo exibido, a função tratadora envia continuamente eventos de usuário para o objeto NCLua (o qual é escalonado na fila de eventos, vide Seção 2). Ao receber um evento de usuário do tipo *refresh* a função tratadora é responsável por chamar a função de atualização de conteúdo (*mediaplayer.updateContent*) para aquele tipo de mídia. Sendo assim, a função tratadora é a responsável por manter a taxa de atualização (*fps*) da exibição da mídia. Como o modelo de execução do NCLua só permite o tratamento de um evento por vez, é importante que as funções retornem o mais rápido possível. Animações, por exemplo, devem ser feitas baseadas em eventos do usuário e não em laços *(loops)* infinitos.

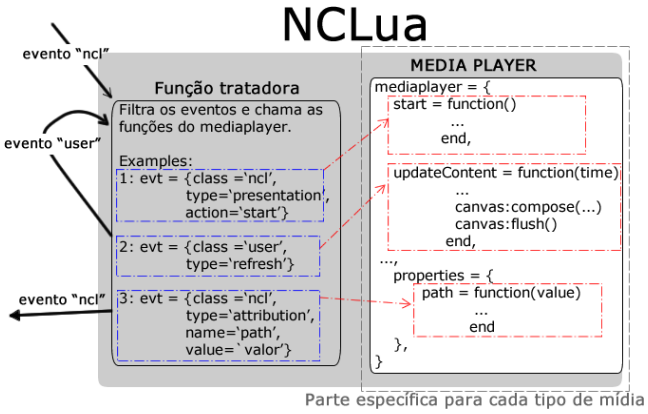

Figura 3. Exemplo de padrão de interface entre documentos NCL e exibidores de mídia baseados em NCLua

Para implementar um exibidor em NCLua baseado nesse modelo padrão de interface, o código NCLua do exibidor deve criar uma tabela cujo nome é *mediaplayer* e especificar as funções relativas aos eventos que deseja tratar. As funções *start*, *stop*, *pause* e *resume* têm, respectivamente, o papel de iniciar, parar, pausar e resumir a exibição da mídia.

As propriedades também são notificadas ao autor do novo tipo de mídia como funções. Assim, caso o novo tipo de mídia, por exemplo, reconheça uma propriedade *fontColor* o autor deve especificar uma função *mediaplayer.properties.fontColor* que será chamada sempre que uma alteração no valor dessa propriedade for recebida pela função tratadora.

O objeto NCLua que representa o exibidor também pode conter âncoras de conteúdo (elemento <*area*>) de tal forma que eventos sobre essas podem ser disparados pelo documento NCL. Ao receber um evento sobre uma âncora, a função tratadora verifica se na tabela *mediaplayer* existem as respectivas funções em *mediaplayer.area*. Como exemplo, caso um evento de apresentação e ação de *start* sobre a âncora de conteúdo "parte1" seja recebida pela função tratadora, ela deve chamar a função *mediaplayer.parte1.start*().

A Listagem 4 apresenta um objeto de mídia referenciando um exibidor NCLua (linhas 1 a 4), com identificador *myplayer*. Esse objeto representa um exibidor para um novo tipo definido segundo o modelo proposto. O elo *initplayer* (linhas 6 a 14) é o responsável por atribuir os valores iniciais das propriedades do exibidor.

```
1:  <media id="myplayer" src="myplayer.lua"
                descriptor="dLua">
2:       <property name="path" />
3:       ...
4:  </media>
5:
6:  <link id= "initplayer"
          xconnector="onBeginSet">
7:
8:    <bind role="onBegin" component="myplayer"/>
9:
10:   <bind role="set" component="myplayer"
            interface="path">
11:     <bindParam name="var" value="path_file"/>
12:   </bind>
13:    ...
14:</link>
```

Listagem 4. Objeto de mídia NCLua para exibidor SubRip e suas propriedades

É importante ressaltar que esse modelo de interação é apenas uma sugestão de como novos exibidores podem ser implementados como objetos NCLua. A principal vantagem em usá-lo é que a função tratadora não necessita ser reimplementada para cada novo tipo de mídia, apenas as funções específicas. Seguindo este modelo ou não, o que fica claro é a extensibilidade da linguagem NCL, que está apta a embutir exibidores para novos tipos de mídia como objetos NCLua.

4. CASO DE USO: EXIBIDOR *SUBRIP*

O conteúdo audiovisual dos vídeos pode não ser suficiente para que sejam compreendidos pela audiência. Alguns fatores como deficiência auditiva, ou mesmo a incapacidade de entender o idioma falado nos diálogos, podem acarretar a incompreensão. Por isso, a exibição de vídeos geralmente está associada a uma mídia de texto como legenda, que informa ao espectador o que está sendo dito ou visto durante a apresentação do vídeo.

As legendas associadas aos vídeos não podem ser tratadas como as mídias de texto convencionais. Isso porque além do texto que é exibido, as legendas devem estar acompanhadas da especificação do intervalo de tempo em que devem ser exibidas. Alguns

formatos de legendas, tais como *SubRip* e *SSA*[7] oferecem uma forma padronizada e simples de criar documentos de legendas.

A norma do SBTVD-T não prevê um tipo de mídia específico para exibição de legendas. Tal lacuna pode ser preenchida por um exibidor de legendas implementado como um objeto NCLua. Nesta seção, é apresentado um exibidor de legendas, codificadas no formato *SubRip* [5]. Esse exibidor é implementado como um objeto NCLua, com o objetivo de validar o que foi exposto sobre extensibilidade de NCL.

Os arquivos no formato *SubRip* são nomeados com a extensão SRT e codificados em texto UTF-8 [8]. Um arquivo *SubRip* é composto por várias legendas. Cada legenda inicia-se com uma linha contendo o número da legenda, seguida por outra linha contendo o tempo de início e fim dessa legenda, ou seja, o tempo que ela deve começar a ser exibida e o tempo que deve desaparecer da tela. As próximas linhas compõem o conteúdo da legenda em si, ou seja, o que irá aparecer na tela. Para delimitar o final de uma legenda, uma linha em branco deve ser informada. Os tempos de início e final são informados no formato: *hours:minutes:seconds,milliseconds* e separados por "-->". As legendas são numeradas de forma crescente, iniciando-se de 1. A Listagem 5 apresenta um exemplo de arquivo SRT.

```
1:  1
2:  00:00:20,000 --> 00:00:24,400
3:  Altocumulus clouds occur between six thousand
4:
5:  2
6:  00:00:24,600 --> 00:00:27,800
7:  and twenty thousand feet above ground level.
```

Listagem 5. Exemplo de um documento SRT

4.1 Implementação

O exibidor de arquivos *SRT*, implementado como caso de uso, baseia-se no modelo de comunicação entre documentos NCL e objetos NCLua apresentado. A função tratadora foi implementada como um módulo a parte, denominado *imediaplayer*, justamente com o objetivo de ser reutilizada por outros exibidores.

O início da execução do exibidor NCLua acontece quando um evento *start* é passado para ele. Quando a função tratadora recebe uma ação de *start*, ela invoca a função *mediaplayer.start()* presente no objeto NCLua. A Listagem 6 apresenta essa função. A única coisa que ela faz é alterar o valor da propriedade *running* do *mediaplayer*. Essa propriedade informa se o exibidor está no estado "ocorrendo" ou "pausado". A função *mediaplayer.pause()* altera o valor dessa propriedade para *false*.

```
1:  mediaplayer = {
2:    start = function ()
3:      mediaplayer.running = true
4:    end,
5:    ...
6:  }
```

Listagem 6. Função *start* do exibidor de SRT

No escopo da implementação do exibidor SRT também foi desenvolvido um módulo auxiliar denominado *srtutil* que abstrai as operações referentes às configurações do exibidor de SRT. O módulo *srtutil* contém, por exemplo, os valores *default* para as propriedades do exibidor. Além disso, *srtutil* também possui funções auxiliares, tais como: um analisador sintático (*parser*) para arquivos no formato SRT; uma função que calcula o tamanho da fonte para a legenda e que a desenha na região do exibidor etc.

A Listagem 7 apresenta a função *mediaplayer.properties.path()*. Responsável por receber a atribuição do valor da localização do arquivo SRT. A função *parseFile(path)*, implementada no módulo *srtutil* é utilizada para fazer a análise sintática e retornar uma tabela *parsed_file* que irá conter as informações, para cada legenda, do tempo de início (*beginTime*), tempo de encerramento (*endTime*) e texto a ser reproduzido (*text*). É interessante observar que, com essa função, o próprio *path* da mídia pode ser alterado em tempo de execução, assim como qualquer outra propriedade. Ao alterar o *path*, o objeto NCLua continua no mesmo ponto de execução, ou seja, a nova mídia continuará a ser exibida do instante em que a outra deixou de ser exibida. Isso permite, por exemplo, que o idioma da legenda seja alterado mesmo durante a exibição do respectivo vídeo.

```
1:  mediaplayer = {
2:  ...
3:    properties = {
4:      path = function(value)
5:                mediaplayer.subtitles =
                                  parserFile(value)
6:              end
7:    }
8:  ...
9:  }
```

Listagem 7. *Parser* para arquivos *SRT*

A função *updateContent* recebe o tempo corrente (controlado pela função tratadora de eventos) e exibe a legenda, caso esteja no momento de sua exibição. O texto da legenda deixa de aparecer se o tempo de fim for atingido. A Listagem 8 apresenta a função *mediaplayer.updateContent(time)*. As funções *hasBegun* e *hasEnded* testam, para a próxima legenda, se os tempos de início e fim, respectivamente, foram atingidos. As funções *draw_subtitle* e *clear_subtitle* têm o papel, respectivamente, de desenhar e apagar a próxima legenda.

```
1:  mediaplayer = {
2:   ...
3:  updateContent = function (time)
4:    if mediaplayer.isPaused() == false then
5:      if mediaplayer.hasBegun(time) then
6:        mediaplayer.draw_subtitle()
7:      elseif mediaplayer.hasEnded(time) then
8:    mediaplayer.clear_subtitle()
9:      end
10:...
11:}
```

Listagem 8. Função tratadora de eventos de usuário que possuem o tempo atual de exibição

A Listagem 9 ilustra o objeto de mídia NCLua com as propriedades *path*, *fontColor* e *fontFamily* que correspondem, respectivamente, à URL do arquivo SRT, cor e fonte dos caracteres exibidos. A única propriedade para a qual o exibidor requer estritamente a atribuição de um valor é *path*. Cor e fonte não necessariamente precisam ser configuradas, pois já possuem valores *default*.

Seguindo o mesmo padrão de interação e a mesma interface (*imediaplayer*) também foi implementado no escopo deste trabalho um exibidor GIF animado [6] – codificação também não reconhecida pela implementação de referência do Ginga-NCL ou pelo SBTVD-T. O exibidor GIF animado segue exatamente os mesmos princípios apresentados e implementa apenas as funções da tabela *mediaplayer*, a função tratadora é reutilizada pelo módulo (*imediaplayer*). A Figura 4 demonstra a execução de uma

apresentação NCL com dois exibidores GIF animados (o mesmo código fonte em dois objetos de mídia NCL) e um exibidor SRT, todos sincronizados, em dois momentos distintos.

```
1:  <media id="srtplayer" src="srtplayer.lua"
                descriptor="dLua">
2:       <property name="path" />
3:       <property name="fontColor" />
4:       <property name="fontFamily" />
5:       ...
6:  </media>
7:
8:  <link id= "initplayer"
          xconnector="onBeginSet">
9:
10:   <bind role="onBegin"
            component="srtplayer"/>
11:   <bind role="set" component=" srtplayer"
            interface="path">
12:     <bindParam name="var" value="file.srt"/>
13:   </bind>
14:   <bind role="set" component=" srtplayer"
            interface="fontColor">
15:     <bindParam name="var" value="white"/>
16:   </bind>
17:  ...
18:</link>
```

Listagem 9. Objeto de mídia NCLua para exibidor *SRT* e suas propriedades

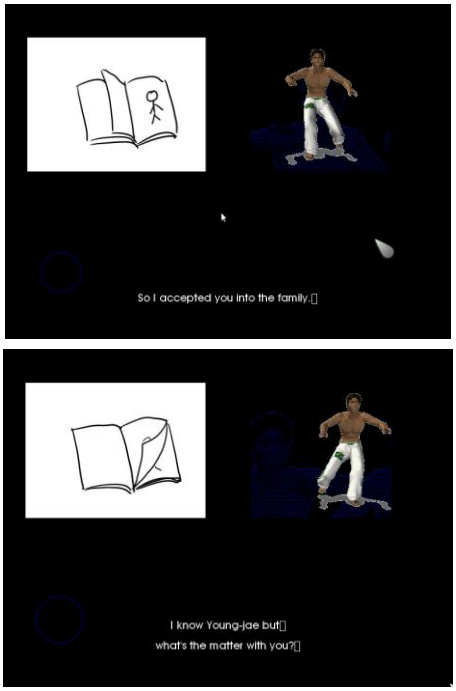

Figura 4. Exibidores SRT e GIF animado implementados como objetos NCLua

5. CONCLUSÕES

Este artigo demonstra como o emprego de objetos NCLua pode estender declarativamente a linguagem NCL. Isso é obtido por meio da especificação de exibidores para novos tipos de mídia. Pelo emprego da proposta deste artigo, é possível exibir mídias em NCL que não foram inicialmente previstas desde que essas mídias possam ser tratadas pela API *Canvas*, usada para exibição em NCLua.

O artigo ainda detalha um exibidor de legendas em formato *SubRip* e um exibidor GIF animado, os quais são propostos como prova de conceito. A implicação desses casos de uso é que autores NCL agora podem definir documentos que incluam legendas ou animações em GIF e fazer seu devido sincronismo de forma declarativa. Esse suporte a novas mídias enfatiza o caráter de NCL como linguagem de cola ao permitir, com base em seu modelo conceitual, o sincronismo mesmo de mídias não-previstas.

Outra contribuição deste trabalho é facilitar a implementação de novos exibidores por meio do padrão de interface proposto. Esse padrão de interface define um formato genérico para a comunicação entre o exibidor NCLua e o documento NCL.

É importante ressaltar, porém, que Lua é uma linguagem de *script* (interpretada) e como tal pode não ser suficientemente eficiente para decodificar todos os tipos de mídia. Sendo assim, dependendo do tipo de mídia em questão, ainda pode ser necessário um suporte nativo do ambiente de execução.

Um trabalho relacionado também à extensão NCL envolve especificar exibidores por meio da execução de comandos de edição ao vivo. A API NCLEdit, interna aos objetos NCLua, permite que o documento NCL em que aquele objeto está inserido possa ser editado durante sua exibição. Como trabalho futuro, esses comandos de edição ao vivo podem ser usados para estender não apenas as mídias suportadas pela linguagem, mas também a semântica de apresentação do documento.

6. REFERÊNCIAS

[1] Soares, L.F.G., Moreno, M.F., Santanna, F. **Relating Declarative Hypermedia Objects and Imperative Objects through the NCL Glue Language**. X ACM Symposium on Document Engineering – DocEng 2009. Munique, Alemanha – Setembro de 2009.

[2] ABNT NBR **Associação Brasileira de Normas Técnicas. Digital Terrestrial Television Standard 06: Data Codification and Transmission Specifications for Digital Broadcasting, Part 2 – Ginga-NCL: XML Application Language for Application Coding**. 2007.

[3] Soares, L.F.G. e Souza, G.L. **Interactive Television in Brazil: System Software and the Digital Divide**. European Interactive TV Conference – EuroITV2007. Amsterdam, Países Baixos – Maio de 2007.

[4] Santanna, F., Cerqueira, R. e Soares, L.F.G. NCLua – **Objetos Imperativos Lua na Linguagem Declarativa NCL**. XIV Simpósio Brasileiro de Sistemas Multimídia e Web – Webmedia 2008. Vila Velha, Brasil – Outubro de 2008.

[5] http://www.matroska.org/technical/specs/subtitles/srt.html, Zuggy, *SRT subtitles*, acessado em julho de 2010.

[6] Compuserve Incorporated. **GIF – Graphics Interchange Format (tm) – A standard defining a mechanism for the storage and transmission of raster-based graphics information**. Compuserve Incorporated, 1987. Disponível em: http://www.w3.org/GRAPHICS/GIF/spec-gif87.txt. Acessado em: 05 jul 2010.

[7] http://www.matroska.org/technical/specs/subtitles/ssa.html, Zuggy, *SSA subtitles*, acessado em julho de 2010.

[8] Yergeau, F. **UTF-8, A Transformation Format of ISO 10646**. *Request for Comments 2279*, Janeiro 1998.