Exemplo 2 - Contador de Cliques

Objetivos:

- Mostrar o uso de Lua como uma extensão à NCL para tarefas de processamento.
- Introduzir o uso de propriedades para integração NCL-Lua.

Comportamento Esperado:

Durante a execução, é exibido o botão *Click It* em quatro momentos diferentes. Se o usuário selecioná-lo com o controle por pelo menos três vezes, ao final da apresentação é exibido o botão *You win*, caso contrário é exibido o botão *You lose*.

Execução:

- Baixe o código do exemplo **aqui**.
- Execute o exemplo e teste os dois finais possíveis.

Considerações:

Esse exemplo mostra o ganho mais básico com a integração NCL-Lua: processamento. Como é sabido, NCL é uma linguagem declarativa focada em sincronismo entre mídias. Não há nenhum mecanismo procedural, como por exemplo, suporte a expressões matemáticas.

Este exemplo também introduz o uso de eventos de atribuição, usados para controlar as propriedades dos nós. A mecânica de uso dos eventos de atribuição é idêntica aos de apresentação, uma vez que compartilham o mesmo modelo de máquina de estados.

Mesmo uma operação básica, como a de incremento, pode ser custosa em NCL. Como fazer em NCL o monitoramento da quantidade de cliques sobre um botão de modo a tomar ações sobre o resultado final? A solução não é tão simples quanto parece pois envolve a criação de muitos elos.

Neste exemplo, utilizando Lua, o procedimento é bastante simples:

- Define-se uma propriedade *inc* no nó NCLua que incrementa um contador sempre que o botão é selecionado (por um elo *onSelectionSet*).
- Ao final da exibição, uma outra propriedade do NCLua, *counter*, é consultada e seu valor decide que botão de fim será exibido (por um elo *onCondGteBeginStart*).

Em relação ao código NCL, algumas considerações:

A definição do nó NCLua deve incluir as duas propriedades:

```
<media id="clicks" src="clicks.lua">
    <property name="inc"/>
    <property name="counter"/>
</media>
```

O elo para incrementar o contador é acionado na seleção do botão, que deve desaparecer:

```
<link xconnector="onSelectionStopSet">
    <bind role="onSelection" component="button"/>
    <bind role="stop" component="button"/>
    <bind role="set" component="clicks" interface="inc">
        <bindParam name="var" value="1"/>
    </bind>
</link>
```

O conector correspondente aceita um parâmetro para o *set*, no exemplo o incremento é de *1*:

```
<causalConnector id="onSelectionStopSet">
    <simpleCondition role="onSelection"/>
```

```
    <connectorParam name="var"/>
    <compoundAction operator="seq">
        <simpleAction role="stop"/>
        <simpleAction role="set" value="$var"/>
    </compoundAction>
</causalConnector>
```

A exibição do fim depende do valor da propriedade *counter*:

```
<link xconnector="onCondGteBeginStart">
    <linkParam name="var" value="3"/>
    <bind role="onBegin" component="video" interface="areaTotal"/>
    <bind role="attNodeTest" component="clicks" interface="counter"/>
    <bind role="start" component="win"/>
</link>
```

O conector correspondente aceita um parâmetro para usar na comparação de *maior ou igual*, que no exemplo vale *3*:

```
<causalConnector id="onCondGteBeginStart">
    <connectorParam name="var"/>
    <compoundCondition operator="and">
        <simpleCondition role="onBegin"/>
        <assessmentStatement comparator="gte">
            <attributeAssessment role="attNodeTest" eventType="attribution"
                                 attributeType="nodeProperty"/>
            <valueAssessment value="$var"/>
        </assessmentStatement>
    </compoundCondition>
    <simpleAction role="start"/>
</causalConnector>
```

O conector e elo para o caso de derrota é bastante parecido e pode ser visto no código fonte.

Conforme mostrado acima, o incremento é acionado por uma *simpleAction* de *set*. Como vimos no primeiro exemplo do tutorial, ações são enviadas aos tratadores de eventos dos NCLua. A ação de *set* é um apelido para uma ação de *start* em um evento do tipo *attribution*, portanto, o evento gerado é o seguinte:

```
evt = {
    class    = 'ncl',
    type     = 'attribution',
    property = 'inc',
    action   = 'start',
    value    = '1',
}
```

Consulte a **referência de eventos NCL** para mais informações.

O código Lua deve controlar a propriedade *counter* que será consultada pelo *assessmentStatement* na decisão do fim. Além disso deve tratar o evento acima.

O código final fica:

```
counter = 0
local counterEvt = {
    class = 'ncl',
    type  = 'attribution',
    property = 'counter',
}

function handler (evt)
    if evt.class ~= 'ncl' then return end
```

```
    if evt.type ~= 'attribution' then return end
    if evt.property ~= 'inc' then return end

    counter = counter + evt.value

    event.post {
        class    = 'ncl',
        type     = 'attribution',
        property = 'inc',
        action   = 'stop'
    }

    counterEvt.value = counter
    counterEvt.action = 'start'; event.post(counterEvt)
    counterEvt.action = 'stop';  event.post(counterEvt)
end

event.register(handler)
```

Repare que os valores das propriedades de um NCLua (ou de qualquer outro objeto de mídia) são controlados pelo formatador NCL. Assim, o NCLua deve notificar ao formatador que o valor da propriedade `counter` foi modificado.

Resumo:

Com este segundo exemplo conseguimos controlar uma variável em um *script* Lua através da ponte de comunicação NCL-Lua. Coube ao NCL solicitar o incremento e testar o valor final da variável. Coube ao NCLua efetuar o incremento sobre a variável, já que NCL não possui suporte a operações matemáticas.

Até aqui é importante que o leitor esteja familiarizado com os seguintes conceitos:

- Eventos NCL e similaridades entre os tipos *presentation* e *attribution*.
- Propriedades não necessariamente estão associadas a variáveis Lua.
- Conectores e elos de atribuição e consulta a propriedades.

Exercícios:

- Fazer o mesmo exemplo em NCL puro.
- Fazer com que quanto mais tarde os botões forem exibidos, maior seja o valor de incremento no contador ao recebimento de um clique.