Exemplo 1 - Modelo de Execução

Objetivos:

- Mostrar o modelo de execução orientado à eventos de NCLua.

Comportamento Esperado:

Ao iniciar o documento NCL, três nós NCLua são iniciados:

- O primeiro é um script vazio (sem nenhuma linha de código). Não há nenhuma inicialização e portanto nenhum tratador de eventos.
- O segundo registra um tratador de eventos que gera um fim natural ao receber um *start*.
- O terceiro registra um tratador de eventos que cria um *timer* de três segundos que por sua vez gera um fim natural ao expirar.

O documento NCL também cria botões que sinalizam o estado de cada NCLua. Associado a cada NCLua há:

- Um elo que inicia um botão verde quando o NCLua é iniciado.
- Um elo que inicia um botão vermelho e pára o verde correspondente quando o NCLua é terminado.

O resultado visual é o seguinte:

- O primeiro NCLua nunca termina e seu botão verde é exibido para sempre.
- O segundo termina imediatamente e seu botão vermelho correspondente é exibido.
- O terceiro exibe o botão verde que após 3 segundos muda para vermelho.

Execução:

- Baixe o código do exemplo **aqui**.
- Execute o exemplo e observe o comportamento dos três nós.
- Leia o código NCL e o três scripts Lua. Abaixo são discutidos os conceitos que envolvem suas implementações.

Considerações:

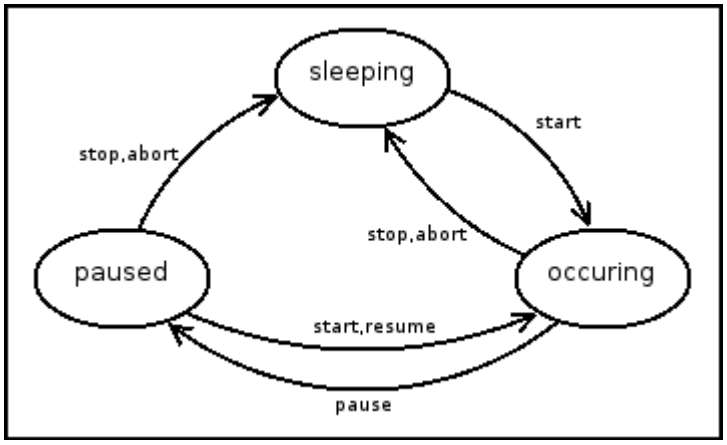
Um NCLua é somente executado a partir de um documento NCL. Mais especificamente, através do disparo de um elo que inicia o nó NCLua no documento.

Um script Lua está, portanto, sujeito ao mesmo ciclo de vida que todas os outros tipos de mídias do documento. No trecho abaixo, retirado do exemplo, vemos que o código para iniciar os três NCLua não é diferente do comum.

```
<port id="entryPoint" component="lua1"/>
<link xconnector="onBeginStart">
    <bind role="onBegin" component="lua1"/>
    <bind role="start"   component="lua2"/>
    <bind role="start"   component="lua3"/>
</link>
```

A comunicação entre o documento NCL e o script NCLua definidos pelo usuário está restrita ao uso de elos convencionais, exatamente como para outros objetos de mídia.

Em NCL, elos operam sob causalidade, ou seja, condições satisfeitas geram ações em resposta. Além disso, as condições e ações que ancoram os elos se baseiam no conceito de *máquinas de estado de eventos*, que em NCL respeitam o diagrama abaixo:



Conforme definido na linguagem NCL, há três tipos de evento: *presentation*, *attribution* e *selection*. Por enquanto vamos nos ater aos eventos de apresentação, que são os explorados neste exemplo. Eventos do tipo *attribution* serão explorados no **exemplo 2**, já eventos do tipo *selection* são tratados pela classe de eventos `'key'` e serão vistos no **exemplo 3**.

No trecho acima, a condição do elo é: *ao iniciar a apresentação do primeiro NCLua*; e a ação é: *inicie a apresentação dos outros dois*. Uma condição é observada e capturada numa transição na máquina de estados, já uma ação ordena uma transição na máquina de estados.

- Condições são passivas, isto é o formatador aguarda a ocorrência da transição para agir. Alguém é, então, responsável em avisar ou notificar o momento da transição, que pode ser o próprio formatador ou o objeto de mídia no caso de transições espontâneas. No último caso, o sentido da comunicação é objeto de **mídia -> formatador**.
- Ações são ativas, isto é, o formatador comanda uma ordem de transição no objeto de mídia que deve proceder conforme o esperado. Nesse caso, o sentido da comunicação é **formatador -> objeto de mídia**.

Com isso em mente, partiremos para a representação dessas idéias em Lua. Como já dito, o modelo de execução de um NCLua é orientado a eventos. Para isso, a especificação do Ginga adicionou a Lua um módulo que dá suporte ao tratamento de eventos, trata-se do **módulo event**.

Todo NCLua deve registrar uma função responsável por tratar os eventos que o formatador aciona. Essa função recebe uma tabela descrevendo tal evento e age conforme desejado. Para o trecho acima que inicia os três NCLua, essa tabela é:

```
evt = {
    class  = 'ncl',
    type   = 'presentation',
    action = 'start',
}
```

O campo `class` denota a classe de eventos, no caso, `'ncl'`. Esse exemplo se restringe à comunicação com o documento NCL, mas há classes de eventos para tratamento do teclado, comunicação com o canal de interatividade, etc. Como era de se esperar, o campo `type` assume um dos dois tipos de eventos NCL e o campo `action`, a ação comandada pelo formatador.

Acabamos de exemplificar o sentido **formatador -> NCLua**, faltando agora o sentido oposto.

Neste exemplo, os dois últimos NCLua notificam seu fim natural ao formatador postando um evento. O código abaixo é comum entre ambos:

```
event.post {
    class  = 'ncl',
    type   = 'presentation',
    action = 'stop',
}
```

É através da chamadas à `event.post` que a comunicação no sentido **NCLua -> formatador** é feita.

Voltando ao exemplo, abaixo são listados os códigos referentes aos três NCLua.

Primeiro NCLua (1.lua):

```
-- vazio
```

Vemos aqui que um código vazio é um programa válido em Lua. Neste exemplo, nenhum tratador de eventos é registrado, portanto, nenhuma ação é tomada e, portanto, esse código jamais sinalizará seu fim natural. O efeito visual é a exibição permanente do primeiro botão verde.

Segundo NCLua (2.lua):

```
function handler (evt)
    event.post {
        class  = 'ncl',
        type   = 'presentation',
        action = 'stop'
    }
end
event.register(handler)
```

Neste caso é registrada uma função tratadora que ao receber um evento (qualquer um), sinaliza seu fim natural. Pelo NCL do exemplo, é fácil deduzir que o primeiro (e único) evento recebido pelo NCLua denota a transição de *start*, ou seja, tão logo é iniciada, a aplicação sinaliza seu fim natural. Visualmente, instantaneamente após a exibição do segundo botão verde, é exibido o botão vermelho correspondente (o botão verde pode não ser visto).

Terceiro NCLua (3.lua):

```
function handler (evt)
    event.timer(3000,
        function()
            event.post {
                class  = 'ncl',
                type   = 'presentation',
                action = 'stop'
            }
        end)
end
event.register(handler)
```

Novamente, uma função tratadora é registrada. No entanto o fim natural não é sinalizado imediatamente ao início da aplicação. Ao receber o evento de início, a aplicação registra um *timer* que após 3 segundos sinaliza seu fim natural. Como efeito visual, temos a exibição do terceiro botão verde e após 3 segundos a mudança para vermelho.

Uma dúvida que pode surgir em relação aos exemplos é a seguinte: Se inclusive o `'start'` inicial chega como um evento ao NCLua, como sei quando o meu script é executado?

A resposta é: você não sabe! A única certeza é que ele é executado antes do recebimento de qualquer evento. O script todo é considerado apenas um inicializador, que deve se preparar para receber eventos conseguintes. Isso também sugere que o seu código inicializador não deve executar ações associadas ao `'start'`, que devem ser movidas para dentro do tratador de eventos.

Resumo:

Este primeiro exemplo explica através de exemplos bem simples a base conceitual do funcionamento de um NCLua. Até aqui é importante que o leitor esteja familiarizado com os seguintes conceitos:

- Eventos NCL, por enquanto apenas do tipo *presentation*.
- Máquina de estados de eventos.
- Comunicação entre o formatador e um NCLua.

Exercícios:

- Usar o explicitDur para forçar término do primeiro NCLua e modificar o documento NCL para que a apresentação seja encerrada.

- Substituir o primeiro NCLua por uma cópia do terceiro e criar um mecanismo de loop entre os dois em que se alternem na apresentação com apenas um deles executando por vez.