Lifia

Introducción a NCL Lua

Conceptos Fundacionales

- Aplicación lua es una “media”
- Aplicación lua tiene multiples “stages”
  - Siempre hay un “current stage”
- Cada Stage agrupa widgets
- El current stage procesa los eventos
- Los widgets pueden ser
  - imagen
  - texto

Codificación en Lua

LUA es un lenguaje:

- Sencillo
- de script con el principal objetivo de ser ligero y extensible
- cuenta con Garbage-collection
- posee un sistema dinámico de tipos

Manual de referencia: http://www.lua.org/manual/5.1/es/

Codificación en Lua

- Definición de variables locales
  Ejemplo: local contador = 0

- Strings
  - se definen entre comillas simples o dobles
    Ejemplo: 'Hola Mundo'
    “Hola Mundo”

  para su concatenación se usan dos puntos
  Ejemplo: 'Hola ' .. 'Mundo'

- Imprimir en la *stdout*
  Ejemplo: print('valor = ' .. 10)

Codificación en Lua

- Definición de funciones

```
function handler(param)
    ...
end
```

- Estructuras de control

```
if cond then … end

if cond then …  else … end

while cond do … end
```

Codificación en Lua

- Sentencia For

```
for i=1, n do
  ...
end
for i=1, n, step do
  ...
end
```

- Estructura de datos: tabla

```
point = {x=10, y=10}
point.x
point["x"]
```

Codificación en Lua

<media>

- src
- Ruta a la aplicación Lua.
- las aplicaciones Lua tienen que tener extensión “.lua”

Ejemplo:

```
<body>

    <media id="canvas" src="canvas.lua" descriptor="descLua"/>

<body>
```

Codificación en Lua

- Módulo Canvas

- Un canvas utiliza la superficie asociada al objeto media Lua para dibujar texto, imágenes, líneas, rectángulos.

- Módulo Event

Permite enviar y recibir eventos desde y hacia la aplicación NCL. Existen 3 tipos de eventos: de teclado, presentation y attribution.

Codificación en Lua

Atributos de Canvas:

- **canvas:attrSize()**
- retorna las dimensiones del canvas.

Ej:

```
local w,h = canvas:attrSize()
```

Codificación en Lua

Atributos de Canvas:

- **canvas:attrColor(color)** función que modifica el color del canvas.

-Ej: canvas:attrColor('white'),
- canvas:attrColor('red')

- **canvas:attrFont(fontFamily,fontSize,fontWeight)** función que modifica atributos de la fuente.

-Ej: canvas:attrFont('vera', 24, 'bold'),
- canvas:attrFont('dejaVuSerif', 20, 'normal')

Codificación en Lua

Funciones:

- **canvas:new(image_path)**
- retorna un nuevo canvas cuyo contenido es la imagen pasada como parámetro.

  - Ej: local img = canvas:new('imagen.png')

Las funciones ya vistas pueden ser aplicadas sobre el nuevo canvas.

Ej: local w,h = img:attrSize()

- **canvas:compose(x,y,src)**

función que compone el canvas principal con el canvas especificado en src en la posición x,y.

- Ej: canvas:compose(0,0,canvas:new('imagen.png'))

Codificación en Lua

Funciones:

- **canvas:drawLine(x1,y1,x2,y2)**

dibuja una línea con sus extremos en (x1,y1) y (x2,y2)

  - Ej: canvas:drawLine(10, 10, 100, 100)

- **canvas:drawRect(mode,x,y,width,height)**

dibuja un rectángulo. El parámetro mode puede tomar los valores 'frame' o 'fill'.

- Ej:
- canvas:drawRect('fill', 10, 10, 100, 100)
  canvas:drawRect('frame', 50, 50, 200, 200)

Codificación en Lua

Funciones:

- **canvas:drawText(x,y,'texto')**

función que dibuja un 'texto' en la posición x,y

- Ej: canvas:drawText(10, 10, 'Hola Mundo!')
  canvas:drawText(5, 40, 'Ejemplo Lua')

- **canvas:flush()**

función para actualizar la superficie del canvas.

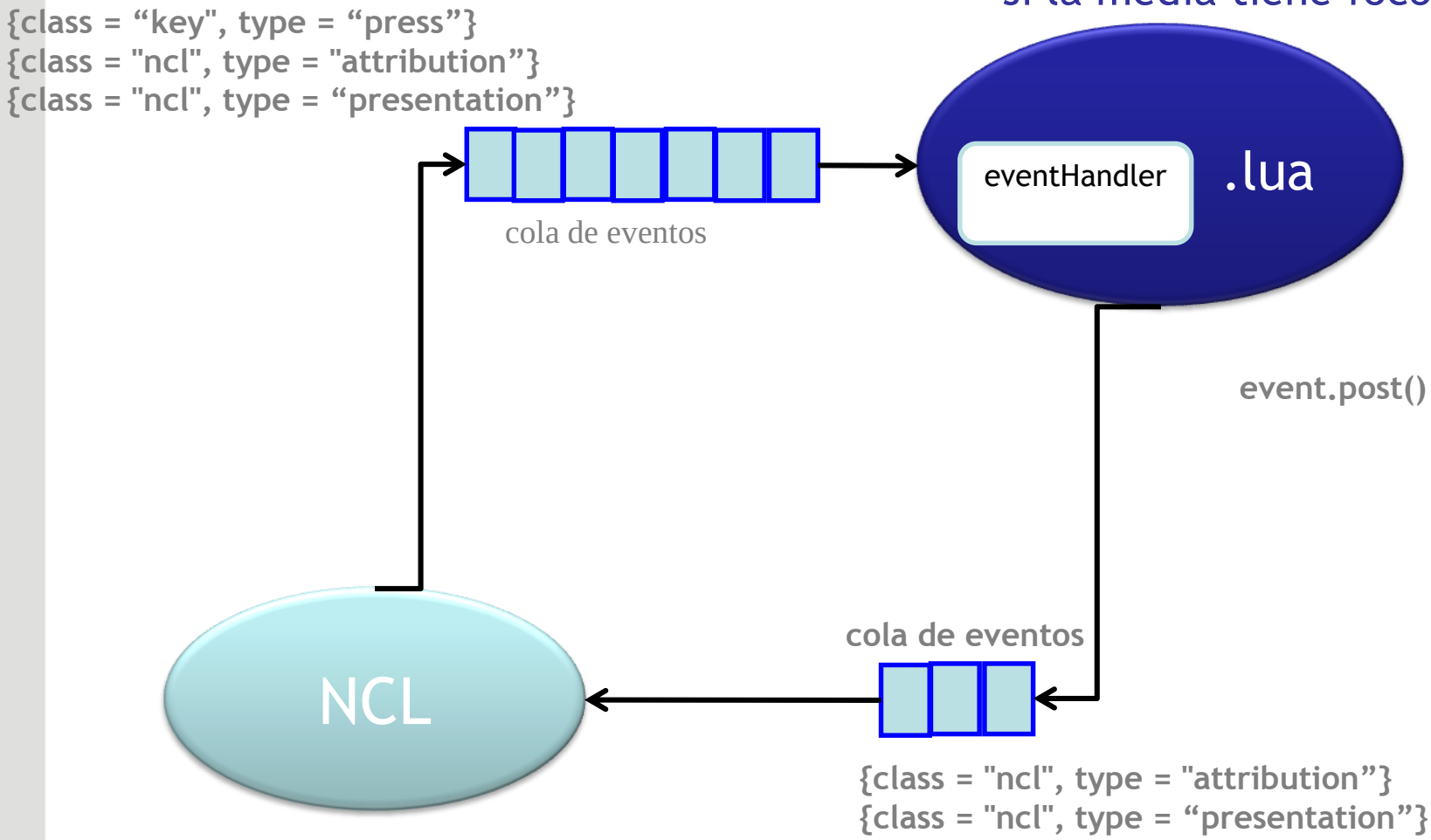
- Ej: canvas:flush()

Ejemplo código Lua

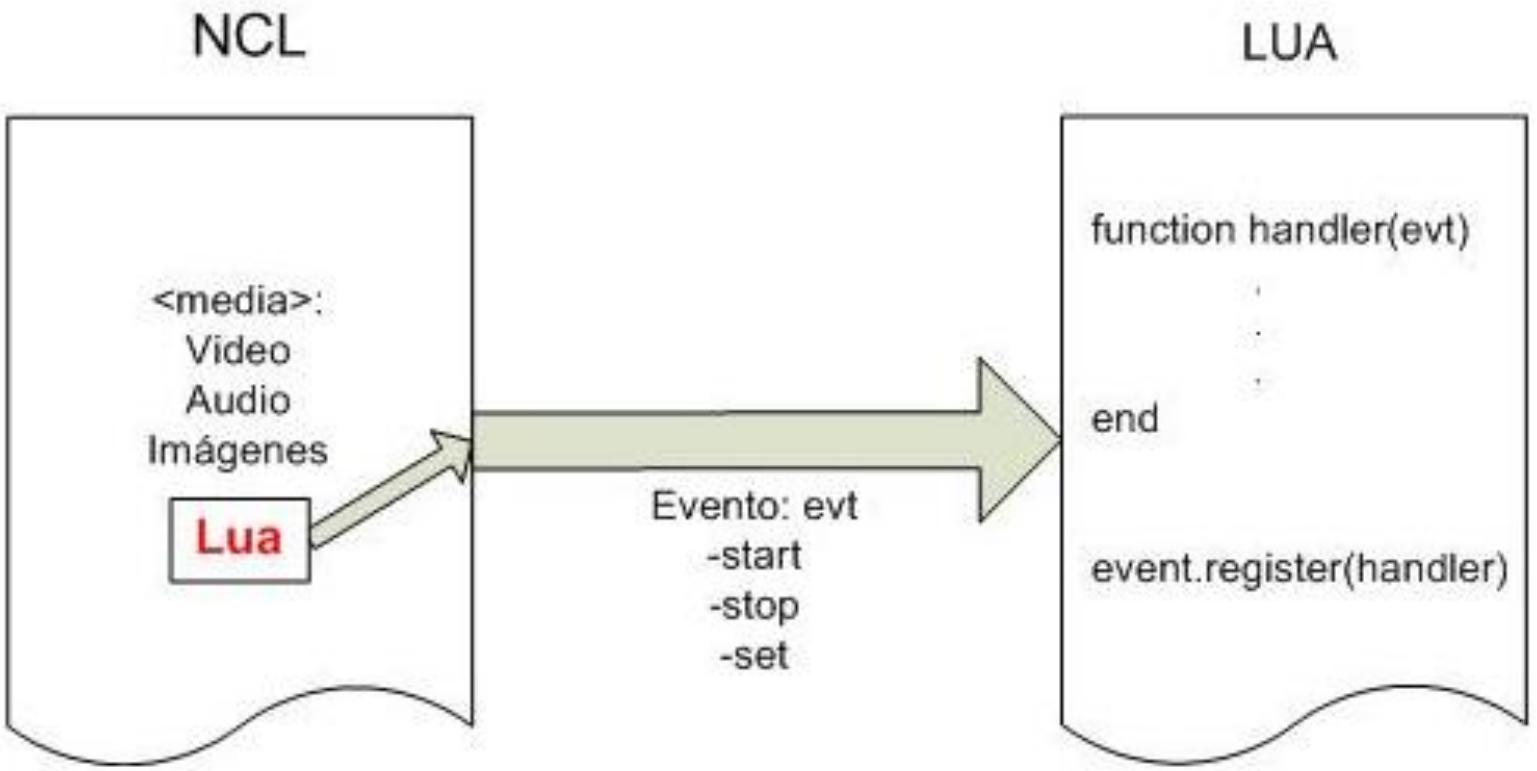
```
canvas:attrColor('blue')
canvas:drawRect('fill',0, 0, 400, 300)
canvas:attrColor('white')
canvas:attrFont ('Tiresias', 24, 'bold')
canvas:drawText(10, 10, 'TV Digital')
canvas:flush()
```

Codificación en Lua

- Lua recibe todos los eventos que ocurren en la aplicación NCL, si la media tiene foco.



- A su vez el script .lua podrá enviar a la aplicación NCL eventos por medio de la función event.post()


NCL
LUA
<media>:
Video
Audio
Imágenes
Lua
Evento: evt
-start
-stop
-set
function handler(evt)
end
event.register(handler)

Lifia

Envio de eventos de NCL a Lua

Codificación en Lua

¿Cómo especificar los objetos Media que reciben las teclas del control remoto?

- Especifiar un **focusIndex** en el descriptor al que se asociará el objeto Media:

```
<descriptor id="descTexto" region="regTexto"  focusIndex="appFocus" />
```

- Asignar a la propiedad **service.currentKeyMaster** el valor del focusIndex de dicho descriptor

```
<media id="foco" type="application/x-ginga-settings">
    <property name="service.currentKeyMaster" value="appFocus"/>
</media>
```

Codificación en Lua

Clasificación de eventos:

- Class = 'key'
  - type: indica el tipo de acción realizada ej: 'press', 'release'
- Class = 'ncl'
  - type: indica el tipo de acción realizada ej: 'presentation', 'attribution', 'selection'

Manejador de eventos

- Un evento se capta mediante una función manejadora de eventos.
- El manejador de eventos tiene como parámetro un evento.

Ejemplo:

```
function handler(evt)
    ...
end
```

Además debemos indicar que el manejador de eventos reciba todos los eventos de la aplicación

```
event.register(handler)
```

Codificación en Lua

Evento key:

- Para captar un evento del control remoto usamos la clase key.

- Estructura básica del evento key

```
Evt = {
    class = 'key',
    type = 'press',
    key = 'F1',   --recordar que es el botón RED
    }
```

- key: indica la tecla presionada

Codificación en Lua

- Ejemplo:

```
function handler(evt)

  if evt.class ='key' and evt.type ='press' then
          if evt.key == 'F1' then
                  ....
          end
  end

end

event.register(handler)
```

Ejercicio

- Implemente una aplicacion NCL que:
  - Haga “echo” de las teclas de colores que el ususario oprime

Envio de eventos de Lua a NCL

event.post()

- Durante su ejecución, un script Lua puede enviar eventos para comunicarse con la aplicación NCL

Por ejemplo:

- que una presentación debe ocurrir
- que se debe modificar una propiedad

- La función event.post efectúa el envío de eventos.

Parte 4 - Codificación en Lua

Evento presentación:

Usado para controlar objetos media desde Lua.

En el ncl:

```
<media id="appLua" src="script.lua" descriptor= "descLua"  >
  <area id="imageArea" label="areaLabel"/>
</media>

<link xconnector="onBeginStart">
   <bind role="onBegin" component="appLua"

  interface="imageArea"/>
  <bind role="start" component="imagen" />
</link>
```

Evento presentation

- Estructura básica del evento presentación:

```
Evt = {
    class = 'ncl',
    type = 'presentation',
    label = string,
    action = string,
      }
```

- Label: indica el nombre del area asociada al evento.
- Action: puede tener los sgtes. Valores 'start', 'stop', 'abort', 'pause' y 'resume'

Parte 4 - Codificación en Lua

En la aplicación Lua:

```
function handler(evt)
  if evt.class =='key' and evt.type =='press' and
                                                evt.key == 'F1' then
          event.post('out',
      {
                  class = 'ncl',
                  type = 'presentation',
         label = 'areaLabel',
                          action =
  'start'
      })
  end
end
```

Evento attribution

- Estructura básica del evento attribution:

```
Evt = {Class = 'ncl',
       Type = 'attribution',
       Action = string,
       Name = string,
       Value = string}
```

- Action: puede tener los valores 'start' y 'stop'
- Name: indica el nombre de la propiedad.
- Value: nuevo valor que se asignará a la propiedad

Codificación en Lua

- En la aplicación NCL:

```
<media id="lua" src="ejemplo.lua" descriptor="desc">
    <property name="srcNuevo"/>
  <!-- A la imagen se le pueden setear parametros,
    por defecto src se refiere al origen de la imagen-->
</media>

<media id="imagen" src="imagen.png" descriptor="otroDesc">
    <property name="src"/>
</media>
```

Codificación en Lua

- En la aplicación NCL:

```
<causalConnector id="onEndAttributionStopSetStart"> <!-- Nombre
  Conector -->
    <connectorParam name="var"/>
    <simpleCondition role="onEndAttribution"/> <!-- Cuando termine una
      region -->
    <compoundAction operator="seq"/>
        <simpleAction role="stop"/>
        <simpleAction role="set" value="$var"/> <!-- Se asigna el valor a
      traspasar -->
        <simpleAction role="start"/>
    </simpleCondition>
</causalConnector>
```

Codificación en Lua

- En la aplicación Lua:

```
function handler(evt)
  local evento = {class = 'ncl',
                  type = 'attribution',
                  Name ='srcNuevo',
                  Value='imagen2.jpg'}

    if evt.class =='key' and evt.type =='press' then
            if evt.key == 'F1' then --boton RED
               evento.action = 'start'
               event.post(evento)
               evento.action = 'stop'
               event.post(evento)
            end
    end
end
```

Codificación en Lua

- En la aplicación NCL:

```
<link xconnector="onEndAttributionStopSetStart">
    <bind component="lua" interface="srcNuevo"
          role="onEndAttribution"/> <!-- Cuando se aprete una tecla -->
    <bind component="imagen" role="stop"/>
    <bind component="lua" interface="srcNuevo“ role="getValue"/> <!--
      Se define el parametro que pasa LUA -->
    <linkParam name="var" value="$getValue"/> <!-- Cuando se coloca
      $getvalue se solicita el valor a LUA y se guarda en var-->
    <bind component=“imagen" interface="src" role="set"/> <!-- El
      pametro src de imagen se setea al valor de var -->
    <bind component="imagen" role="start"/>
</link>
```

Es necesario recuperar el valor asignado desde .lua (getValue) en una variable de ncl (var) para hacer el bind con la propiedad a modificar (src)

Codificación en Lua

```
<causalConnector
  id="onEndAttributionStopSetStart“>
    <connectorParam name="var"/>
    <simpleCondition
      role="onEndAttribution"/>

    <compoundAction operator="seq"/>
        <simpleAction role="stop"/>

       <simpleAction role="set“
      value="$var"/>




       <simpleAction role="start"/>
    </simpleCondition>
</causalConnector>
```

```
<link
  xconnector="onEndAttributionStopSetStart">

    <bind component="lua"
      interface="srcNuevo“role="onEndAttribut
      ion"/>

    <bind component="imagen" role="stop"/>

    <bind component="lua"
      interface="srcNuevo“ role="getValue"/>
      <linkParam name="var"
      value="$getValue"/>
     <bind component=“imagen"
      interface="src" role="set"/>

    <bind component="imagen" role="start"/>
</link>
```