

Aplicações interativas para a TV digital brasileira

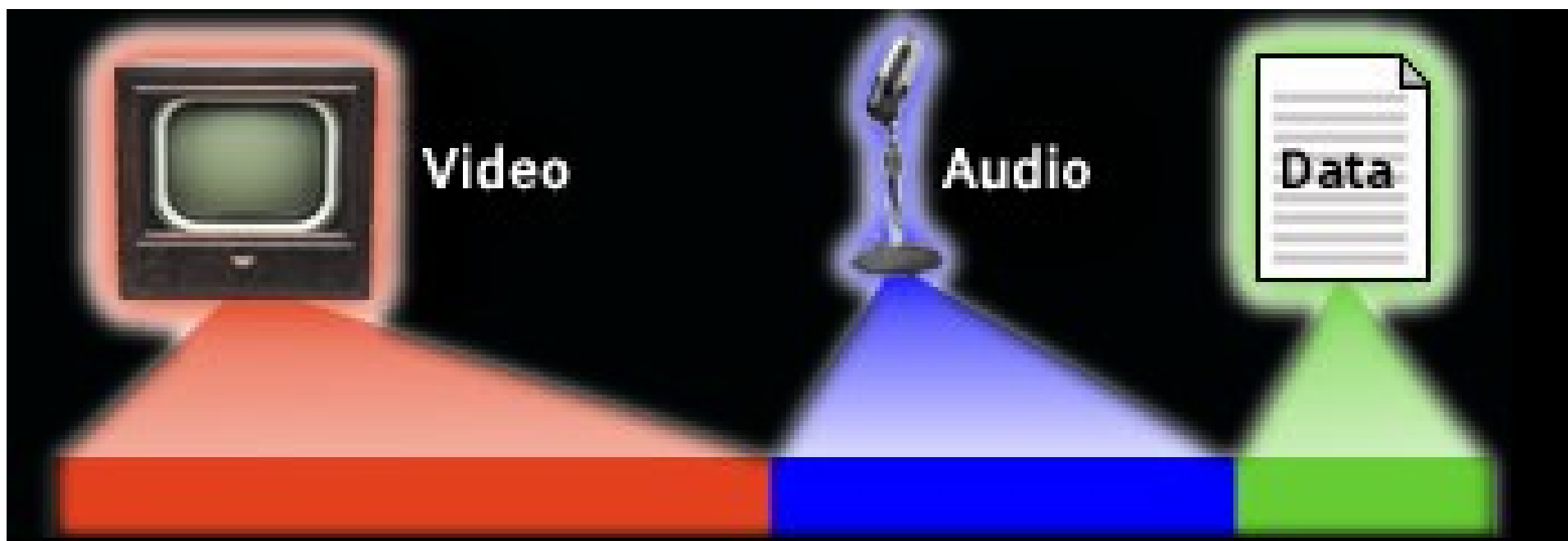
Francisco Sant'Anna

PUC-Rio

Laboratório Telemídia



Digital TV



NCL – Nested Context Language

- XML language for multimedia synchronization
 - scripted with *Lua*
- Focus on interactive applications for Digital TV:
 - Declarative language for the Brazilian system
 - ITU-T recommendation for IPTV services

A simple example

```
<ncl>
  <port component="main" />

  <media id="main" src="main.mpeg">
    <area id="main_info" begin="30s" />
  </media>
  <media id="info" src="info.png" />

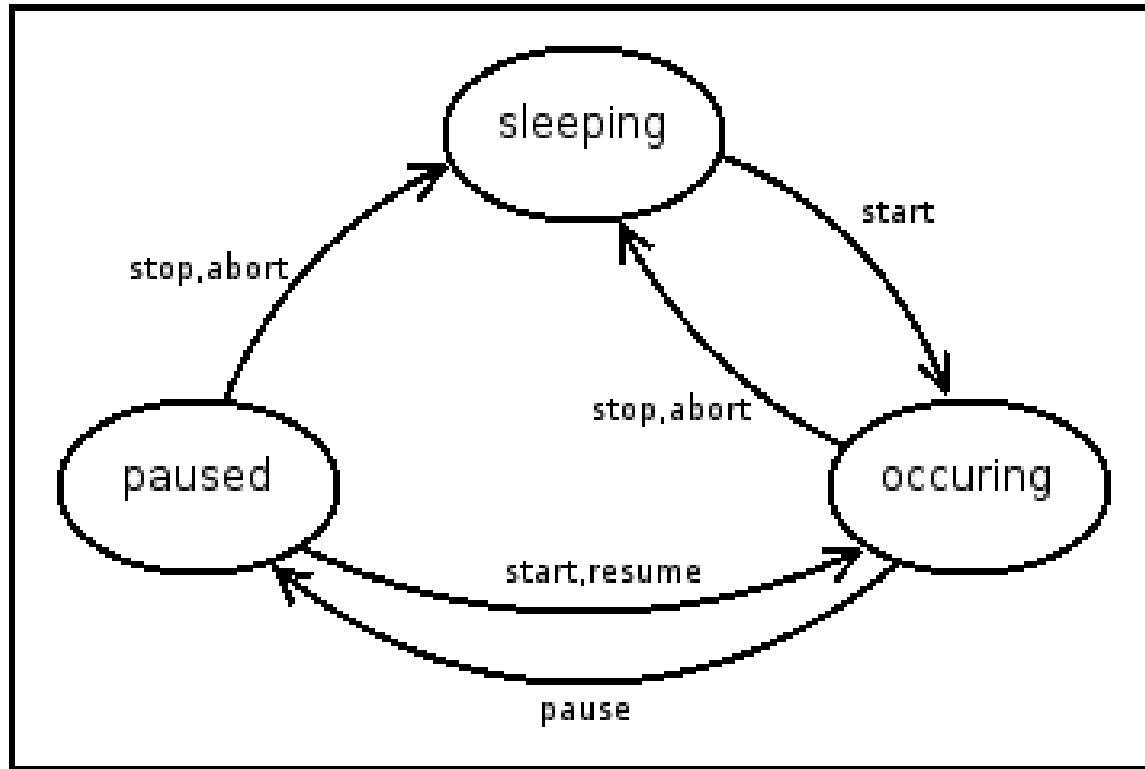
  <link>
    <bind role="onBegin"
          component="main"
          interface="main_info" />
    <bind role="start"
          component="info" />
  </link>
</ncl>
```

Objects Interfaces

- Used as synchronization points
- Content Anchors:
 - information units (frames, pixels, words, ...)
 - `<area>` element
- Properties:
 - (transparency, duration, sound level, ...)
 - `<property>` element

Relating Objects

- Transitions on interfaces' state machines



A simple example

```
<ncl>
  <port component="main"/>

  <media id="main" src="main.mpeg">
    <area id="main_info" begin="30s"/>
  </media>
  <media id="info" src="info.png"/>

  <link>
    <bind role="onBegin"
          component="main"
          interface="main_info"/>
    <bind role="start"
          component="info"/>
  </link>
</ncl>
```

NCL + Lua = **NCLua**

- Requirements:
 - Few modifications in the languages
 - Strict borderline between them
 - different teams
 - Orthogonal integration
 - *(no side-effects)*

NCLua

- Are <media> nodes
 - Complete code separation
- Communicate through events
- Have no access to the document structure
- Have custom semantics for its interfaces
 - properties and areas

Additional Lua modules

- event
 - Bi-directional communication
- canvas
 - Graphical primitives

Event-driven paradigm

- Main Loop
- Dispatcher
- Queue
- **Handlers**
- **Generator**
- **Events**

Event-driven paradigm

- Only one event is handled at a time
- Processing must be fast
- Inversion of control
- Lua is appropriate:
 - tables
 - closures
 - co-routines

The event module

- `event.register(f)`
 - `function (evt) ... end`
- `event.post(evt)`
- `event.unregister(f)`
- `event.timer(ms, f)`

Event classes

- Communication with NCL
 - class: 'ncl'
 - type: 'presentation', 'attribution'
 - action: 'start', 'stop', 'abort', ...
 - label/name: 'fim', 'fase1', 'counter', ...

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

Event classes

- Remote control
 - class: 'key'
 - type: 'press', 'release'
 - key: 'RED', 'A', '1', ...
 - { class='key', type='press', key='RED' }
 -

Example 1 – Execution model

- Three NCLua nodes are started
 - The first does not handle events
 - The second notifies its natural end when started
 - The third creates a 3 seconds timer and notifies its natural end
- Buttons to identify NCLua's states

Example 2 – Clicks counter

- The “Click it” button appears on screen
- We want to count the number of times the user clicks the button
- In pure NCL: exponential number of links
- With Lua: a property for the counter

Example 3 – A simple game

- Interaction with the remote control
- The user should move the monkey to the banana
- The button “You win” is shown on screen

Canvas module

- The global `canvas` represents the NCLua region
- `canvas:new(img_path)`
- `canvas:attr*()` `-- Size, Color, Font`
- `canvas:draw*(x,y)` `-- Rect, Line, Text`
- `canvas:compose(x, y, other)`

Conclusion

- Minimalist API, basic primitives
 - Specification is small and succinct
 - Small implementation
- Mechanism vs Policy
 - Does not impose a programming style
 - Abstractions in pure Lua
 - Players and native applications in Lua

Thanks!



<http://www.telemidia.puc-rio.br>

<http://www.ncl.org.br>