Time Evaluation for the Integration of a Gestural Interactive Application with a Distributed Mulsemedia Platform

SALEME, Estêvão Bissoli, CELESTRINI, J. R., and SANTOS, C. A. S.
Federal University of Espírito Santo, Brazil
Outline

- Introduction and motivation
- Experimental environment
  - Distributed mulsemedia platform
  - Gestural Interactive application
  - Test setup, design (procedure)
- Results
- Discussion and conclusion
Intro and Motivation

- Synchronization of **MulSeMedia (Multi-Sensory Media)** has to do with **transmission, production, and presentation** of multiple signals (EG et al., 2015)

- For some sensory media, such as **olfaction**, the delay is acceptable for a few seconds (GHINEA et al., 2010; MURRAY et. al, 2013; MURRAY et. al, 2016) whereas others, such as **visual and haptic**, it can be stricter (MILLER, 1968; CARD et al., 1983; NIELSEN, 1993; YUAN et al., 2015)

- **PlaySEM (distributed mulsemedia platform)** -> has nearly no synchronization loss with **timeline applications** (SALEME et al., 2015)

- **Delay is unknown** when working with **other kinds of applications** (event-based) so far
Our research

- We integrated the PlaySEM platform with a gestural interactive application presented by Santos et al. (2015), which is a simulation of a theatrical play where the presenter interacts through gestures to perform actions or sensory effects on the stage.

  - How long does the computational time (for transmission and production) of this integration take considering the distributed architecture?
  - What happens before the interactive application calls the sensory effects? What happens after that?
  - What can we do to improve it when necessary?
Our scenario

1. User
2. Gestural interactive app.
3. PlaySEM SE Renderer
PlaySEM

- The PlaySEM (Sensory Effects Metadata) platform consists of a set of open source tools (based on SeSim) written in Java for playing and rendering sensory effects in the user environment.

- 3 main components: SE Video Player, SE Renderer, and a microcontroller.

(SALEME et al., 2015)
# PlaySEM’s architecture

## Applications

SE Player, interactive apps, VR/AR apps, SEMP, Sensible Media Sim, VLC, MIT Sensory Fiction, etc

## PlaySEM

<table>
<thead>
<tr>
<th>SE Renderer</th>
</tr>
</thead>
<tbody>
<tr>
<td>UPnP bus interface</td>
</tr>
<tr>
<td>SetSem(String mpegvMetadata); SetPlay() ...</td>
</tr>
</tbody>
</table>

## Devices

PlaySEM hardware, Philips amBX, Vortex, CAVE System, Olorama Aromatizing, etc

MPEG-V processing

Devices’ driver
Interactive application

- Interactive spaces ⇒ well-defined physical environment composed of heterogeneous devices, applications, and users
- Santos et al. (2015) developed an interactive application in which a user interacted through gestures with a system to control the scenes of a theatrical play producing lighting, wind and vibration effects
- Other examples of interactive environments:

  MEDIATE, multisensorial interactive environment for children with autism
  Source: http://www.annacarreras.com/eng/mediate/

  The Exhale pavilion
  Source: https://www.e-architect.co.uk/miami/exhale-pavilion
Interactive application

- **3W1H (Where, What, Why, How) approach** for developing interactive applications (REHEM NETO et al., 2015)
- Gap: ad-hoc and unstructured solutions
- It formalizes a set of stages for developing interactive scenarios
- Based on the principle of stimuli-response
- It combines actions that change the scenario with specific events are recognized
Interactive application

- Gestures, actions and their relationship

How will it happen?

1. Context Transitions

2.

3.

(SANTOS et al., 2015)
Interactive application

Behavior Chain (Rules) as an XML file

Excerpt of an action (MPEG-V script)

(SANTOS et al., 2015)
Test setup

- Arrangement of devices, tools and network

User Environment

- User
- Ultrabook
  - Kinect
  - 192.168.1.10
- Mini-pc
  - Actuators
  - 192.168.1.15
- Wireshark
- Running the Interactive application
- Running the SE Renderer
Experimental Procedure

![Experimental Procedure Diagram]

1. Recognition
2. Packaging
3. Transmission
4. Metadata processing
5. Execution
Results

Before the sensory effects

1. **Gesture recognition** takes around **10ms** (processing)
2. **Packging MPEG-V scripts** takes from **13 to 28ms** (processing)
   - Increase in delay when having more actions - **UPnP request**
   - Send all scripts as soon as possible and afterward just call for action(s) -> it would **remove packaging** time (it requires changes in the SE Renderer)

<table>
<thead>
<tr>
<th>Behavior Frame</th>
<th>Gesture</th>
<th>Actions</th>
<th>Recognize (ms)</th>
<th>SendSem (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>BF02</td>
<td>Raise right hand</td>
<td>ACT02</td>
<td>9,39</td>
<td>13,21</td>
</tr>
<tr>
<td>BF03</td>
<td>Keep right hand to right side</td>
<td>ACT03</td>
<td>9,80</td>
<td>13,76</td>
</tr>
<tr>
<td>BF04</td>
<td>Keep left hand to left side</td>
<td>ACT04</td>
<td>9,76</td>
<td>13,33</td>
</tr>
<tr>
<td>BF05</td>
<td>Raise left hand</td>
<td>ACT05</td>
<td>9,47</td>
<td>12,73</td>
</tr>
<tr>
<td>BF06</td>
<td>Raise both hands</td>
<td>ACT05, ACT08, ACT09</td>
<td>10,45</td>
<td>27,70</td>
</tr>
<tr>
<td>BF07</td>
<td>Keep right hand to right side</td>
<td>ACT06</td>
<td>11,76</td>
<td>17,76</td>
</tr>
<tr>
<td>BF08</td>
<td>Keep left hand to left side</td>
<td>ACT07</td>
<td>10,19</td>
<td>15,28</td>
</tr>
</tbody>
</table>
Results

- Transmitting and processing
  1. **Transmission** time is affected by the **type of network** as expected
     - **SetSem** conveys data; **SetPlay** is just a command
     - Send all scripts after handshaking -> it would suppress the command **SetSem** for transmission as well as for processing
  2. **Processing** is not expensive; it reaches up to 10ms
     - It could be optimized replacing the programming language
Results

- Range of average time (wired network) = 27ms to 67ms
Discussion and Conclusion

- The range of 27ms to 67ms reveals the average time taken for dealing with computational aspects before presenting an effect.

- Previous time + devices’ activation time + perceptual time must be considered when developing a distributed mulsemedia solution.

- Aspects to be observed for improving global response time:
  - Avoid sending MPEG-V metadata to the platform in real-time
  - Reduce the number of messages exchanged on the network
  - Consider using a wired network (test setup)
  - Use a more efficient programming language to convert MPEG-V metadata into commands for sensory effect devices

- Future work: an infrastructure for mulsemedia solutions concerning integration processes, software, and hardware inspired by CPS.
References

Ragnhild Eg, and Carsten Griwodz. 2015. **Approaching New Limits of Synchrony with Multi-Sensorial Media.** IEEE MMTC R-Letter 7-9, 10/2015, IEEE


Questions?

Estêvão Bissoli Saleme
estevaobissoli at gmail.com

http://dx.doi.org/10.1145/3083187.3084013