Developing Responsive and Interactive Environments with the ROSS Toolkit

Andrea Bellucci
Universidad Carlos III de Madrid
Avenida de la Universidad, 30
28911, Leganés, Madrid, Spain
abellucc@inf.uc3m.es

Ahmed Sabbir Arif
Synaesthetic Media Lab
Ryerson University
Toronto, Ontario, Canada
asarif@ryerson.ca

Aneesh P. Tarun
Synaesthetic Media Lab
Ryerson University
Toronto, Ontario, Canada
aneesh@ryerson.ca

Ali Mazalek
Synaesthetic Media Lab
Ryerson University
Toronto, Ontario, Canada
mazalek@ryerson.ca

Abstract

TEI researchers/designers are often discouraged from building complex interactive environments by the requirement of high technical knowledge. In this studio/workshop, we present the ROSS Toolkit that offers tools to abstract/automate low-level programming of technical details, thereby simplifying the design and programming of interactions between heterogeneous networked devices. Participants will be first introduced to the toolkit functionality to cope with different technical issues. In a second part, they will experiment with the toolkit by developing use cases of increasing complexity that will involve off-the-shelf devices, interactive surfaces, and custom-made tangible artifacts. We expect participants to learn what are the opportunities and challenges for the development of responsive and interactive environments.

Keywords

Toolkits; API; interactive environments; prototyping.

ACM Classification Keywords

H.5.2. Information Interfaces. User Interfaces – input devices and strategies, prototyping.

Introduction

Programming responsive and interactive environments is a complex task that requires extensive knowledge of hardware and software components. As a result, TEI

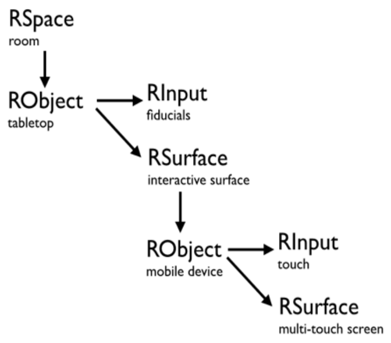


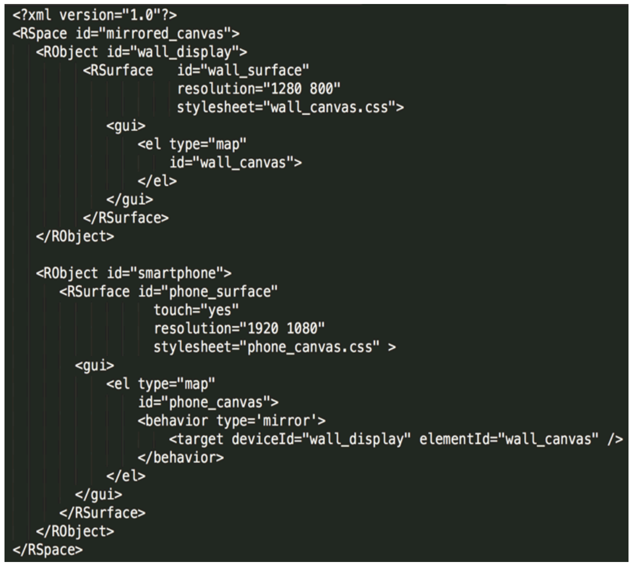
Figure 1. An example of the nested hierarchical structure: a mobile device tracked on an interactive surface to implement a "peephole" display and provide an additional information layer.

researchers/designers are often hindered in building rich and engaging user experiences. Tangible toolkits can support the development of novel designs, especially in a context where not only the software interface matters but also the hardware components, physical affordances, and spatial relationships between devices are important.

In today's increasingly technologically cluttered landscape, we are faced with operating a multitude of technological devices on a daily basis; it is therefore critical for the TEI community to investigate toolkits that can ease the development of applications that run across a variety of platforms and devices [2, 3]. To this end, we developed the ROSS (Responsive Objects, Surfaces and Spaces) Toolkit as a complete re-design of the original ROSS API (Application Programming Interface) [5] specifications to take into account opportunities and needs that have emerged with recent technical developments (e.g., newer microcontrollers, wearables, and sensing technologies).

The ROSS Toolkit

ROSS Toolkit exposes an API as well as tools for abstracting low-level communications between heterogeneous devices and reducing the programming effort for sensor-based and spatial interactions. It provides a conceptual framework that allows the design of interactive environments as hierarchical nested structures. Every device, screen, sensor in an interaction space is mapped within a hierarchical structure, outlined in an XML descriptor file. This hierarchical tree (Figure 1) encapsulates relationships between various entities and determines how they interact. The XML-based document (Figure 2) forms the basis for generating the application code and managing the communication between different sensors and devices.



```xml
<?xml version="1.0"?>
<RSpace id="mirrored_canvas">
   <RObject id="wall_display">
         <RSurface   id="wall_surface"
                     resolution="1280 800"
                     stylesheet="wall_canvas.css">
            <gui>
                <el type="map"
                    id="wall_canvas">
                </el>
            </gui>
         </RSurface>
   </RObject>

   <RObject id="smartphone">
      <RSurface id="phone_surface"
                   touch="yes"
                   resolution="1920 1080"
                   stylesheet="phone_canvas.css" >
        <gui>
            <el type="map"
                id="phone_canvas">
                <behavior type='mirror'>
                    <target deviceId="wall_display" elementId="wall_canvas" />
                </behavior>
            </el>
        </gui>
      </RSurface>
   </RObject>
</RSpace>
```

Figure 2. An example of XML descriptor file.

Together with XML-based authoring, ROSS Toolkit provides: (1) an API packaged as JavaScript components within the Node.js environment. This is exposed to the developers through the XML descriptor and JavaScript functions, (2) a parser to generate the source code (from the XML descriptor) to run the interactive environment, (3) a server running on Node.js (a cross-platform runtime environment) that hosts the client applications, registers the connected clients, and lastly performs the role of a connection and communication bridge between different devices, and (4) client applications that are served onto a device's browser environment and are mostly made up of

JavaScript and CSS files. A client's UI is first rendered as a Jade file (an intermediate HTML template language for applications layout). In the case of Arduino, deployable and Arduino-compatible code is generated by the parser. ROSS uses and builds upon the TUIO protocol to simplify cross-device communication.

Duration

The duration for this Studio/Workshop (S/W) will be one day.

Proposal

The S/W focuses on the design, development and deployment of responsive and interactive environments by researchers/designers with different backgrounds and levels of programming expertise. During the S/W we will introduce the different features of the ROSS Toolkit that support (a) **novice programmers** by providing high-level entry points for tailoring ready-made templates of default application scenarios, and (b) **experienced programmers** by allowing real-time editing of the auto-generated source code.

The first part of the S/W will be broken down into two activities. In the first one, a brief introduction will be given on the concept of responsive and interactive environments: the state of the art of toolkits that allow to increase the capabilities of the physical environment with digital information processing will be presented and major technical challenges will be highlighted, with special attention to the integration of heterogeneous devices that work together. Following this, participants will be introduced to the ROSS Toolkit: (1) its hierarchical structure, (2) how it promotes the programming of multi-device and spatial-aware interactions, (3) the automatic generation of the application source code through parsing an XML descriptor file, (4) the Node.js server environment, and (5) the JavaScript APIs to program advanced functionality. Participants will be guided through a step-by-step explanation of the different stages of the process to build and setup an interactive environment with the ROSS Toolkit.

In the second part of the S/W, participants will be organized in groups and they will make use of the ROSS Toolkit to implement use cases of incremental difficulty, focusing on interactions with digital maps on surfaces: e.g., remote multi-device interaction via canvas mirroring or nearby multi-device interaction for "peephole" behaviors (Figure 3). At the end of the implementation, benefits and drawbacks of ROSS Toolkit will be discussed with participants.

No specific programming skills are required for this S/W. Programming experience with JavaScript, Node.js and Arduino is recommended for participants who want to implement advanced functionality.

Topics to be covered

The Studio will cover the following topics:

- **Design goals for tangible toolkits**: what we need in order to simplify the development of responsive and interactive environments. Reflections on: abstraction, power in combination, inclusivity, low threshold/high ceiling, breadth of API coverage, extensibility.

- **Device ecosystems:** how to integrate the different off-the-shelf devices, interactive surfaces and custom tangibles that coexist in the same

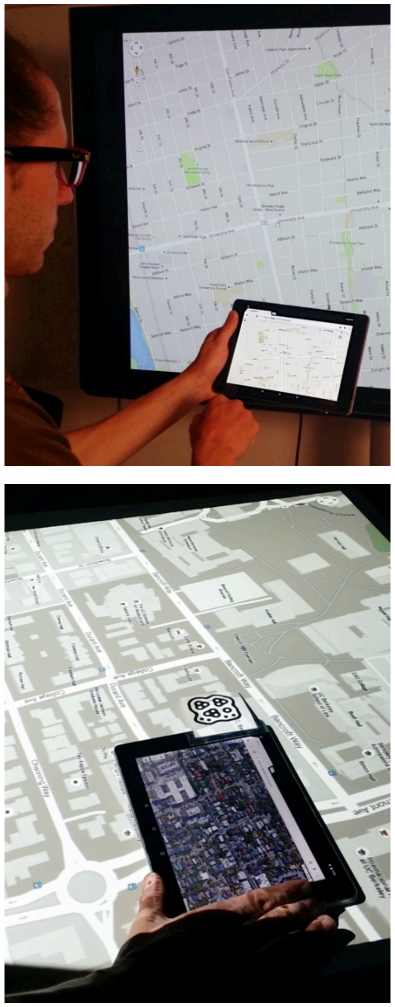


Figure 3. Examples of applications developed with the ROSS Toolkit: canvas mirroring (top) and peephole display (bottom).

space to support human activities. State of the art and open issues.

- **The ROSS Toolkit**: its architecture, features and APIs for reducing the programming effort for sensor-based and spatial interactions. ROSS Toolkit offers: an interactive environment powered by Web applications, hierarchical nested structure, spatial mapping of devices, different levels of abstractions in terms of application behavior, widgets, sensor mapping, and communication.

- **Hands-on implementation**: how the toolkit could be used by TEI researchers/designers for prototyping a variety of meaningful interactions.

Learning Goals

Participants are expected to gain a complete overview of the development of responsive and interactive environments: opportunities and challenges. They will learn how the heterogeneity of devices, platforms and networking technologies makes prototyping a challenging task. They will also experience that, by lowering the technical threshold of prototyping multi-device interactions, the ROSS Toolkit offers the possibility to leave behind low-level details and focus on how best to combine devices to achieve a desired type of tangible user experience. We expect this S/W to be the first opportunity for participants to create, in a short amount of time, full functional prototypes that rely on the synergy of different off-the-shelf and custom devices.

Positioning

Toolkits are expected to nurture the creativity of researchers/designers by providing building blocks that make it easier for them to materialize their creative thoughts [1]. The experimentation with the ROSS Toolkit will shed light on what kind of meaningful interactions are favored/precluded by its idiosyncratic features (re-thinking software in physical space) and the low-threshold/high ceiling trade-off [4]. We will be able to investigate how the rationale of such a toolkit influences and shapes users' creative thinking and design process.

Acknowledgements

This work has been supported by the Canada Research Chairs program and 2015 UC3M postdoctoral mobility scholarship.

References

[1] Greenberg, S. 2007. Toolkits and interface creativity. *Multimedia Tools and Applications* 32, 2, 139–159.

[2] Greenberg, S. and Fitchett, C. 2001. Phidgets: easy development of physical interfaces through physical widgets. In *Proc. UIST '01*. ACM, 209-218.

[3] Marquardt, N., Diaz-Marino, R., Boring, S., and Greenberg, S. 2011. The proximity toolkit: prototyping proxemic interactions in ubiquitous computing ecologies. In *Proc. CHI '11*. ACM, 315–324.

[4] Myers, B., Hudson, S. E., and Pausch, R. 2005. Past, present, and future of user interface software tools. *ACM Trans. Comput.-Hum. Interact* 7, 1, 3–28.

[5] Wu, A., Mendenhall, S., Jog, J., Hoag, L. S., and Mazalek, A. 2012. A nested API structure to simplify cross-device communication. In *Proc. TEI '12*. ACM, 225- 232.