Interactive design of multimodal user interfaces

Reducing technical and visual complexity

Werner A. König · Roman Rädle · Harald Reiterer

Received: 24 July 2009 / Accepted: 22 January 2010 / Published online: 19 February 2010
© OpenInterface Association 2010

Abstract In contrast to the pioneers of multimodal interaction, e.g. Richard Bolt in the late seventies, today’s researchers can benefit from various existing hardware devices and software toolkits. Although these development tools are available, using them is still a great challenge, particularly in terms of their usability and their appropriateness to the actual design and research process. We present a three-part approach to supporting interaction designers and researchers in designing, developing, and evaluating novel interaction modalities including multimodal interfaces. First, we present a software architecture that enables the unification of a great variety of very heterogeneous device drivers and special-purpose toolkits in a common interaction library named “Squidy”. Second, we introduce a visual design environment that minimizes the threshold for its usage (ease-of-use) but scales well with increasing complexity (ceiling) by combining the concepts of semantic zooming with visual dataflow programming. Third, we not only support the interactive design and rapid prototyping of multimodal interfaces but also provide advanced development and debugging techniques to improve technical and conceptual solutions. In addition, we offer a test platform for controlled comparative evaluation studies as well as standard logging and analysis techniques for informing the subsequent design iteration. Squidy therefore supports the entire development lifecycle of multimodal interaction design, in both industry and research.

Keywords Multimodal user interfaces · Post-WIMP user interface · Natural interaction · Design environment · Zoomable user interface · Semantic zooming · Multimodal interaction · Squidy

1 Introduction

With recent advances in computer vision, signal processing, and sensor technology today’s researchers and interaction designers have great opportunities to go far beyond the traditional user interface concepts and input devices. More natural and expressive interaction techniques, such as tangible user interfaces, interactive surfaces, digital augmented pens, speech input, and gestural interaction are available and technologically ready to be incorporated into the multimodal interface of the future (see some examples in Fig. 1). However, the actual utilization of these techniques for the design and development of multimodal interfaces entails various critical challenges that interaction designers and researchers have to face.

In contrast to the design of traditional graphical user interfaces, the development of multimodal interfaces involves both software and hardware components [12]. However, conventional development environments (e.g. MS Visual Studio/Net, Adobe Flash, Eclipse IDE) fall short of supporting uncommon interaction modalities and appropriate data processing (e.g. computer vision), not to mention the handling of multipoint and multi-user applications (e.g. for multi-touch interaction). As a consequence a broad variety of very heterogeneous and specialized toolkits and frameworks have evolved over the last few years such as...
Fig. 1 Diverse input devices for single-modality or multimodal interfaces: (a) Physical game controller offer absolute pointing, motion sensing and gesture-recognition to the end-user. (b) Digital pens build upon users’ pre-existing knowledge and thus offer a very natural mode of interaction e.g. for digital sketching and prototyping. (c) Multi-touch surfaces augmented with physical tokens reduce the gap between real-world and digital-world interaction. (d) Finger gestures provide a very expressive and direct mode of interaction. (e) Well-known devices such as an omnipresent laser pointer provide flexible input from any distance.

Table 1 Interaction designers have to cope with very different environments for the same interaction modality, touch input

<table>
<thead>
<tr>
<th>Hardware platform</th>
<th>Microsoft Surface</th>
<th>Custom-build table</th>
<th>Apple iPhone</th>
<th>HTC Hero</th>
</tr>
</thead>
<tbody>
<tr>
<td>Form factor</td>
<td>Table</td>
<td>Table</td>
<td>Mobile</td>
<td>Mobile</td>
</tr>
<tr>
<td>Operating system</td>
<td>Microsoft Windows</td>
<td>Linux/Windows</td>
<td>Mac OS X</td>
<td>Android OS</td>
</tr>
<tr>
<td>Programming language</td>
<td>C#</td>
<td>C++</td>
<td>Objective-C</td>
<td>Java</td>
</tr>
<tr>
<td>Software framework</td>
<td>Surface SDK</td>
<td>Touchlib</td>
<td>iPhone SDK</td>
<td>Android SDK</td>
</tr>
</tbody>
</table>

Apple iPhone SDK\(^1\), Microsoft Surface SDK\(^2\), GlovePIE\(^3\), Processing\(^4\), NUI Group Touchlib\(^5\). They provide support for specific interaction modalities, but are mostly restricted to a dedicated hardware environment and entail further requirements and dependencies. When using touch as input for instance, the interaction designer has to cope with different hardware platforms, operating systems, programming languages, and software frameworks (see Table 1). When developing single-modality interfaces, this diversity can be bypassed—at least in the short-run—by focusing on just one specific device. But the combination of multiple devices, e.g. for multimodal interaction involves further platforms, devices, and frameworks, resulting in an unmanageable technical and mental complexity.

There are development environments that support at least some of the more uncommon input devices and modalities (e.g. physical turntables, mixing desks, multi-touch surfaces and simple vision tracking). Two examples are Max/MSP\(^6\) and vvvv\(^7\). Both are graphical development environments for music and video synthesis and are widely used by artists to implement interactive installations. Their popularity in the design and art community arises in particular from their graphical user interface concepts. Both are based on the concept of visual dataflow programming and utilize a cable-patching metaphor to lower the implementation threshold [24] for interactive prototyping. Users arrange desired components spatially and route the dataflow between the components by visually connecting pins instead.

---

\(^3\)GlovePIE, [http://carl.kenner.googlepages.com/glovepie/](http://carl.kenner.googlepages.com/glovepie/).
\(^7\)vvvv, [http://vvvv.org/](http://vvvv.org/).