

INTERACTION-KIOSK FOR OPEN HUMAN-COMPUTER INTERACTION WITH PERVASIVE SERVICES

Markus Eisenhauer, Andreas Lorenz, Andreas Zimmermann

Fraunhofer Institute for Applied Information Technology

Schloss Birlinghoven

53754 St. Augustin, Germany

{markus.eisenhauer, andreas.lorenz, andreas.zimmermann}@fit.fraunhofer.de

Abstract

Traditional computer applications understand mouse and keyboard input for controlling behaviour of the system. Wireless technology allows the same devices for controlling applications from a distance. Depending on goal, task and situation of the user we envision controlling the application with other input devices that fit much better to the user's personal attributes. To make the vision true, we have developed innovative tools for smart combination of the application logic with software libraries handling the device connection and data transmission. With our solution, developers and users are enabled to control applications with any input device, ranging from Joystick, Wii or other game controllers, to mobile phones and PDA, up to web cam with gesture recognition. To support developers to build own solutions we deliver a toolset enabling them to implement interactive applications. The tools facilitate the integration of further innovative input devices in existing and new developed software environments. For our demonstration we integrated the control into a game application. The user can choose an input device according her personal preferences and control the game with that device.

1. Introduction

For the development of ambient services it is unrealistic to base user-computer interaction on traditional mouse or keyboard. In a world of ambient services, the technology disappears into the surroundings until only the user interface remains perceivable by users. Most preferably, the interface to interact with an ambient service is separated from the device hosting the services, e.g. the user interface is running on a mobile device and connected with the devices in the environment.

Iftode et al [1] identified the need for a simple, universal solution to control different applications in the environment of the user, which end-users are likely to accept easily. The remote device should be programmable and support dynamic software extension for interaction with additional embedded services. Using for instance the *Personal Universal Controller* [2], a user can speak the name of a command through which this is executed by the system. The focus here is on automatic creation of the user interface from a service description language. Head-tracker solutions like [3] are designed to work with gestures for replacing traditional pointing devices. Using a Web-cam, it allows users to point and click by simply aiming their face. A combination of pointer position and keystroke

input device is described in [4], using miniature video cameras that track finger position where the user can type or point in the air.

The goal is to enable the developer to be able to test any device with any application, without the need to install specific drivers etc. The developer can easily connect any device and provide users the experience to test the device that fits best to the current situation, personal capabilities and gusto. In particular the use of devices the user is already familiar with could be transferred and tested in additional or new services. To support system development we provide a toolkit for developing software solutions to enable any device as input device delivering meaningful events to services. The toolkit abstracts from concrete input devices focusing on transmission of meaningful events without constraints to the shape of the device or interaction modality. In the back-end the toolkit uses common Web-Service technology to remotely operate the service over the network.

2. Realisation

By principle, software applications react on input commands independent of the device being used to create it. More important than the concrete device is to distinctly define the *meaning* of a command, and that the user is able to anticipate the triggered reaction of the computer application.

For the connection of input devices we use an approach for distributed system development based on adapted web service technology. With this approach, we precisely distinguish between client and server parts of the system. On both sides we carefully avoid installation of large backend systems. We took particular care that our approach does not require the installation and maintenance of a complete web server on the server side. Our toolkit comprises all necessary components for the definition of input events to be processed, the communication between input devices and application hosts, and the attachment of input listeners to services on the server. The software transmits input commands over wireless networks (WLAN, Bluetooth), and distributes incoming events to the operating system or the selected processing applications. The development of specific device drivers shades off and end users are enabled to choose whatever input device best fits to their personal preferences.

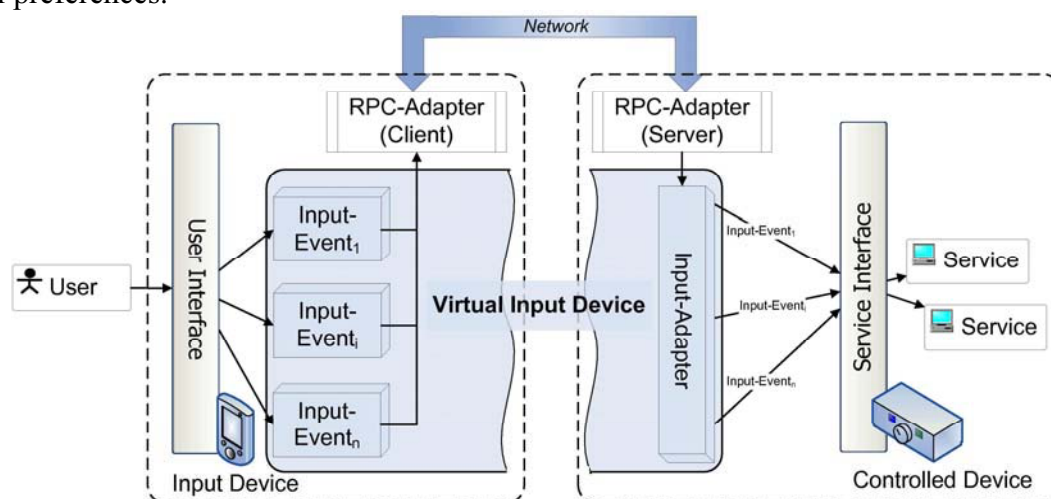


Figure 1 The abstract system architecture

Figure 1 shows the abstract system architecture. In the middle of the image, a *Virtual Input Device* is the source of meaningful user input without constraints for physical shape of the device, type of user interface or interaction style.

3. Demonstration

For the demonstration of the Interaction-Kiosk we developed a small system running on a stand alone computer. The computer is hidden from the public and only the display is visible. The visitor of the exhibit is offered to select any of seven different interaction devices for controlling the graphical system on the display. For illustration purposes, we will run a *PacMan* game in full screen mode. The user's task is to play the game with the chosen device, i.e. to control the small yellow icon moving in the four directions "Up", "Down", "Left" and "Right". At any time, the user can select any other device without interruption of the game. Potentially, all devices can be used in parallel to control the application. Because the game does not provide a multi-player mode, we restrict the number of active players to one. Because the demonstration is similar to a kiosk of interaction styles, we call it "Interaction-Kiosk" in our work.

Devices

The Interaction-Kiosk provides a set of different input devices mapped to the standard input events of keyboard input and mouse pointer control. The set of devices is composed of mobile and handheld devices, game-controllers and gesture-recognition systems.



Figure 2 The setup of the Interaction-Kiosk

Visitors of the kiosk have the opportunity to select any device from the following list:

- Navigation buttons of an infrared remote control (i.e. Microsoft's Remote Control for Windows XP Media Center Edition 2005)
- Rocker switch or dragging on the touch-sensitive display of a PDA (by finger or stick)

- Dedicated game controller: Joystick and rocker switch of a XBox-Controller, a Wii-Controller
- Dance-Mate
- Camera-based gesture recognition of aiming a pen with attached retro-reflecting marker

Figure 2 illustrates the kiosk, with the dancing mate at the lower right corner and the other devices on the table below the *PacMan*-screen (from left: PDA, pen, Wii, XBox, and remote control).

Procedure

Visitors of the demonstrator were first provided with a general introduction to our research work. Because we cannot expect the visitor to have proper background knowledge, we provide detail explanation about the background of the project. During the game play we also provide help, assistance and feedback if requested by the user. If necessary, we will show how to create specific control events, e.g. be demonstrating the gestures.

The left picture of Figure 3 shows a user employing a stick for controlling the game. On top of the screen you see the infrared camera tracking the reflector surface attached to the end of the pen, which is passive and only reflecting the flash of the camera in the figure. The user is aiming with the pen in the air for making the *PacMan* moving to the intended direction. In the right picture of the figure, another user is doing similar gestures on the screen of the PDA. By dragging with pen or finger on the display, the user controls the *PacMan* as if drawing a line to the intended direction. The MDA transmits the derived control commands to the server using Wireless-LAN.



Figure 3 Examples of using gestures to control the *PacMan* game

4. References

- [1] L. Iftode, C. Borcea, N. Ravi, P. Kang, and P. Zhou. Smart phone: An embedded system for universal interactions. In 10th IEEE International Workshop on Future Trends of Distributed Computing Systems, 2004.
- [2] J. Nichols, B. A. Myers, M. Higgins, J. Hughes, T. K. Harris, R. Rosenfeld, and M. Pignol. Generating remote control interfaces for complex appliances. In 15th annual ACM symposium on User interface software and technology, Paris, France, 2002. ACM Press.
- [3] R. Kjeldsen. Improvements in vision-based pointer control. In Assets '06: The 8th international ACM SIGACCESS conference on Computers and accessibility, pages 189-196, New York, NY, USA, 2006. ACM.
- [4] F. Ahmad and P. Musilek. A keystroke and pointer control input interface for wearable computers. In 4th IEEE Conference on Pervasive Computing and Communications, pages 2-11, 2006.