Spinner: A Simple Approach to Reconfigurable User Interfaces

Shigeru Kobayashi
International Academy of Media Arts and Sciences
3-95, Ryoke-cho, Ogaki City
Gifu, Japan 503-0014
mayfair@iamas.ac.jp

Masayuki Akamatsu
International Academy of Media Arts and Sciences
3-95, Ryoke-cho, Ogaki City
Gifu, Japan 503-0014
aka@iamas.ac.jp

ABSTRACT

This paper reports our recent development on a reconfigurable user interface. We created a system that consists of a dial type controller 'Spinner', and the GUI (Graphical User Interface) objects for the Max/MSP environment[1]. One physical controller corresponds to one GUI controller on a PC's display device, and a user can freely change the connection on the fly (i.e. associate the physical controller to another GUI controller). Since the user interface on the PC side is running on the Max/MSP environment that has high flexibility, a user can freely reconfigure the layout of GUI controllers. A single 'Spinner' control device consists of a rotary encoder with a push button to count rotations and a photo IC to detect specific patterns from the GUI objects to identify. Since 'Spinner' features a simple identification method, it is capable of being used with normal display devices like LCD (Liquid Crystal Display) or a CRT (Cathode Ray Tube) and so on. A user can access multiple 'Spinner' devices simultaneously. By using this system, a user can build a reconfigurable user interface.

Keywords

Reconfigurable, Sensors, Computer Music

1. INTRODUCTION

How to access many parameters (e.g. various parameters of a digital keyboard synthesizer or software synthesizer) by limited number of physical controllers (e.g. knobs, sliders, switches and so on) has been a long-term issue for digital musical instruments. Typically, making controllers assignable is a solution. For examples, many MIDI controllers equip assignable physical controllers on their surface, and users can configure their settings of connections from a physical controller to a parameter (or multiple parameters)[2][3]. This seems to be a popular approach to associate physical controllers to a lot of GUI controllers or parameters. However, there is a drawback with this kind of approach: It is hard to remember a connection between a controller and a parameter.

To solve this issue, some solutions have been provided:

- Put a template sheet for each controller to remember what parameter is assigned to the controller (e.g. microKONTROL from KORG[4]).
- Equip a small LCD for each controller to show what parameter is assigned to the controller (e.g. Nord Modular G2 from Clavia DMI[5]).

These kinds of approach are rather effective, but still have drawbacks: Since the layout of physical controllers is fixed, a user still needs to do a conversion from a position of the physical controller in a physical space to the position of a virtual controller in a virtual space. This conversion might be improved with practice, but still it requires long time to do it automatically.

One possible solution to this issue is making a user interface reconfigurable. For example, LEMUR from JazzMutant makes a suggestion[6]. LEMUR features a touch-panel (with multi-touch capabilities) based controller, so a user can configure the desired layout of GUI objects and access them via the touch-panel. This is a good example of the possibilities of reconfigurable user interfaces. But there are still drawbacks with a LEMUR type of approach:

- Requires a dedicated display device with multi-touch capabilities, so choice is limited.
- No appropriate tactile feedback that corresponds to type of controllers (e.g. knobs, sliders, switches and so on) are provided.

As we discussed, every physical controller has some sort of drawback in terms of binding a physical controller to a parameter of synthesis, so we tried an approach to create a reconfigurable user interface by a simple method. We call the first dial type of prototype 'Spinner'.

As a dial (or knob) type of interface, Audiopad[7] is a well known work. Spinner takes a different approach from Audiopad. In Audiopad, a dial (puck) is much more than a physical dial (for instance, it is used to select a sample to play, set value of a parameter and so on). In Spinner, a dial is just a simple physical dial. We intended to keep the dial as simple as possible.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
NIME05, Vancouver, BC, Canada
Copyright 2005 Copyright remains with the author(s).

2. BASIC CONSTRUCTION PRINCIPLES

The basic idea of 'Spinner' is as follows:

- One dial for one parameter (a physical dial corresponds to a GUI dial on the PC's screen).
- Virtually, a user can access many dials simultaneously.
- Assigning a dial to a parameter is simply done by pushing the knob of the dial.
- No special display device is needed and works with usual display devices like LCD, CRT, PDP (Plasma Display Panel) and so on.

Figure 1 shows a basic usage of 'Spinner'. One dial is connected to the interface device, and then the interface device is connected to the PC. A user can access a parameter on the PC's screen by rotating the dial. If the person wants to access another parameter, he can just lift the dial off from the display device, then put it on another GUI object (virtual dial) and push the knob to start an identification process. In the identification process, a special pattern for identification is displayed on the display device, and the dial device detects where it is located.

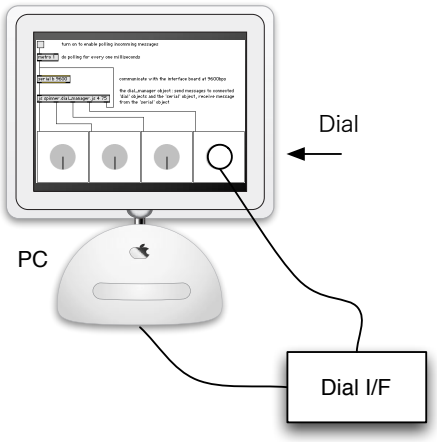

Figure 1: One dial is associated with the GUI dial that is where the physical dial is located on.

Figure 2 shows the typical transaction between a PC side and a dial interface side. At first, the dial interface sends a request to start identification, then the PC side starts the identification process. During the identification period, the PC side sends the current index number to the dial interface side just before showing the identification pattern. When all indexes have been iterated, the PC side and the device side both exit the identification period. Once the identification period finished, the interface device sends commands with the new ID.

3. IMPLEMENTATION AND RESULTS

3.1 Materials

The 'Spinner' system consists of the following components (only main components are listed):

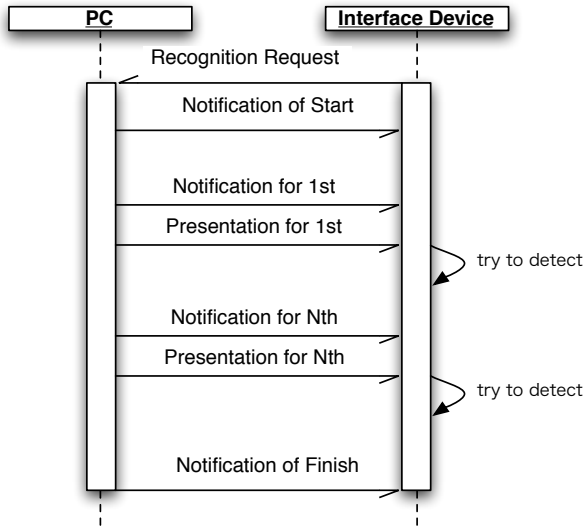

Figure 2: Typical transaction between a PC side and an interface device side.

- PC side
  - Dial manager object (spinner.dial_manager.js)
  - GUI dial object (spinner.dial.js)
- Dial interface device side
  - Interface board
    - dsPIC30F4011 (Microchip)
  - Dial
    - Rotary encoder EC11E1834404 (ALPS Electric)
    - Photo IC S7184 (Hamamatsu Photonics)

Figure 3 shows actual basic blocks of the first prototype. The interface board is connected to PC via the USB connection. The dial devices are connected to the interface board via the special cable.

3.1.1 PC side implementation

PC side components are programmed on Max/MSP environment. Both dial manager and GUI dial object are written in Javascript. An instance of dial manager ('spinner.dial_manager') handles all messages from/to interface board via 'serial' object. Instances of GUI dial object ('spinner.dial') act as GUI version of dial. Figure 4 shows an example screenshot.

3.1.2 Dial side implementation

Figure 5 shows components of a dial device. The rotary encoder is a popular rotational sensor which turns continuously, and outputs a sequence of digital pulses[8]. All signals from rotary encoders are processed by a dsPIC[9] on the interface board. dsPIC is a microcontroller from Microchip, which combines a 16-bit microcontroller (MCU) with a digital signal processor (DSP). In the first prototype, we use only the MCU part of the dsPIC, but we have a plan to use the DSP part to improve accuracy of identification.

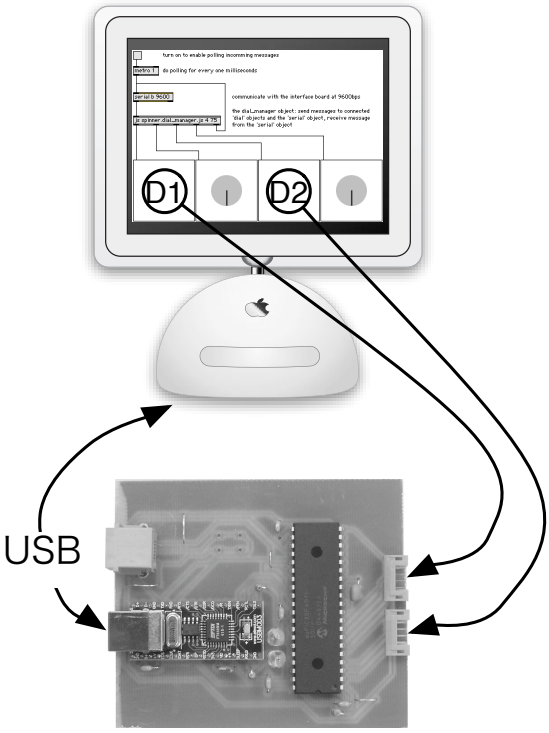

Figure 3: An example of setup. Two dial devices are connected to the interface board, and the interface board is connected to the PC via USB. Since the interface board is USB bus-powered, the setup will be simple.

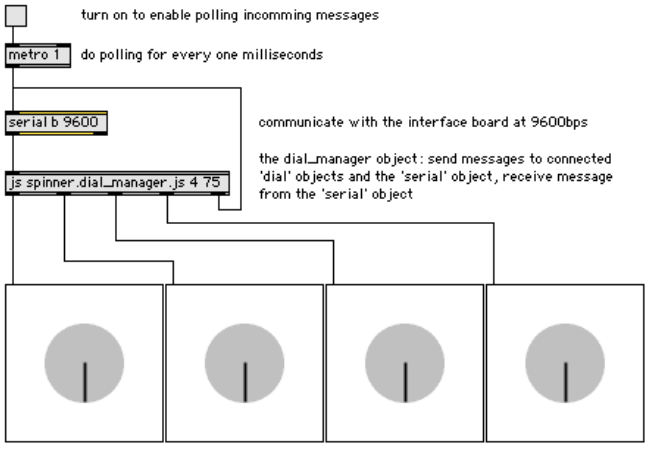

Figure 4: Four 'spinner.dial' objects are connected to a 'spinner.dial_manager' object. Since 'spinner.dial' object is configurable (color, shape, size and so on), a user can easily make variations. A user can make deeper changes by modifying the Javascript code.

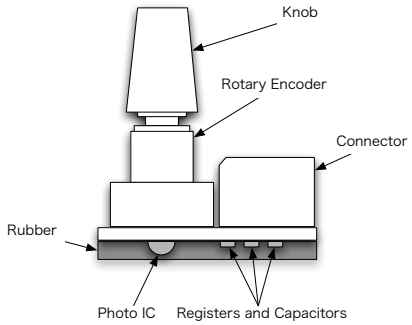

Figure 5: Side view of a dial device. The dial has a push button, and a photo IC is mounted on the bottom face.

As shown in figure 5, a photo IC is mounted on the bottom face of a dial to detect what color is presented on a display where the dial is located on.

3.2 Identification

In the first prototype, the method of identification is very simple. Figure 6 shows the timing chart of the identification process.

In each period, a message showing a current index is sent, and a pair of identification patterns are presented. The messages are sent from the PC side to the dial I/F side via USB. The identification patterns consist of black or white.

Index messages are sent to the physical dial and the corresponding dial is painted black. Other dials are painted white. If the physical dial detects thet the GUI dial beneath it is black, it looks at the last index message sent to know with which GUI dial it is associated.

For example, a message that indicates "current dial should be 2" is sent and a physical dial detects "black", this means that the physical dial is located on the second GUI dial object, so the dial will be associated with the second GUI dial.

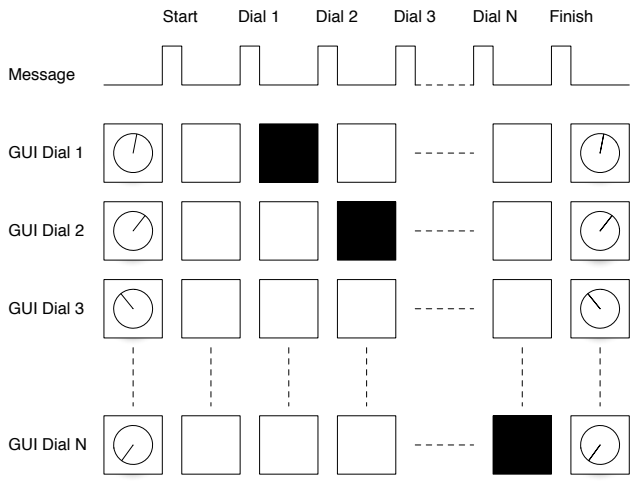

Figure 6: The timing chart of an identification process.

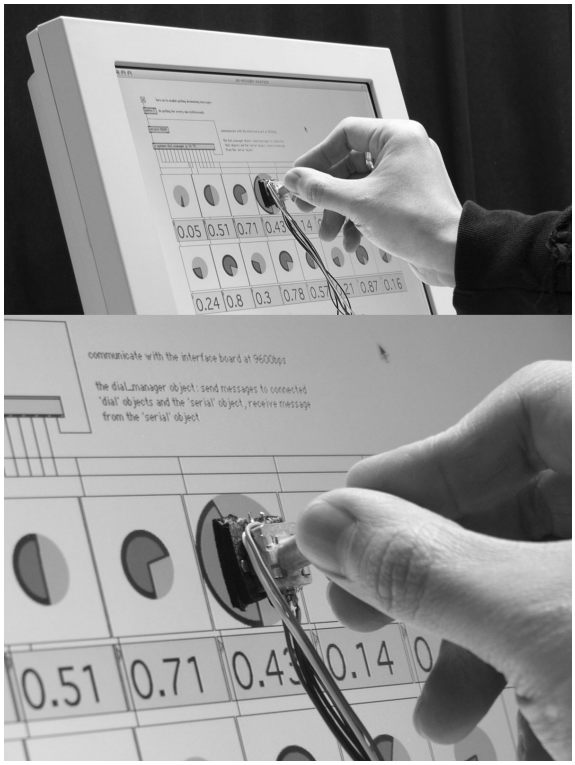

Figure 7: A user put the dial on the screen. The associated GUI dial is enlarged to let a user see the current value around the physical dial.

3.3 Results

Figure 7 shows an example of how the first prototype is used. The user puts the dial on the screen. He can change the value of the associated GUI dial by rotating the knob of the physical dial. He can freely change the connection simply through 'move and push' action. the associated GUI dial is enlarged to let the user see the current value. We found that it is easy to use. In regard to this point, we saw many performance possibilities. However this version has several drawbacks.

We have tested the first prototype with several LCD panels. Recently, most new LCD panel products tend to be thinner, and as a result, the surface of many LCD panels are weak and it is a little difficult to use 'Spinner' with such kind of LCD panels. For such kind of LCD panels, we had to put a thin protection panel that is made of transparent acrylic on it.

For the identification process, we had to adjust the threshold level manually before using our dial, since the proper white and black threshold level varies for each LCD display. After adjustment, the identification process worked as expected.

For identification time, 75ms is required per one GUI dial, For example, if a user instantiate 4 'spinner.dial' objects, about 300ms is required for the identification process. This might be not so big a problem if a user accesses few parameters and does not change locations of physical dials frequently. However, we should find a solution for this issue.

One possible solution is replacing the PC side with dedicated front-end application instead of Max/MSP environment. Since the PC side is implemented in Max/MSP using Javascript, the accuracy of timing for presenting identification patterns (white or black) is not so good. If we replace this part with a dedicated application written in OpenGL[10], we will be able to present identification patterns at higher rate. In addition, if we employ a more sophisticated blinking pattern, the identification time will be shorter. Our goal is under the typical reaction time[11].

4. CONCLUSIONS AND FUTURE WORK

We developed the first version of 'Spinner'. It is satisfactorily usable and we can feel many possibilities. However, this version has limitations. In the next stage, we would like to improve 'Spinner' as follows:

- Wireless connection via WirelessUSB[13], Bluetooth[14], ZigBee[15], etc.
- Faster and more accurate identification with automatic calibration.
- More sophisticated design of physical dial.
- More sophisticated error handling.
- Other types of controllers (e.g. switch, slider).

5. ACKNOWLEDGMENTS

This research was a part of the 'dspbox' project[12]. The 'dspbox' project was a yearly research project by Masayuki Akamatsu of IAMAS. Many thanks to Katsuhiko Harada and Kazuki Saita (students, Institute of Advanced Media Arts and Sciences) for creating prototypes of 'Spinner'.

6. REFERENCES

[1] Cycling '74. Max/MSP. http://www.cycling74.com/, 2005.
[2] EDIROL. http://www.edirol.com/, 2005.
[3] M-AUDIO. http://www.midiman.net/, 2005.
[4] KORG. microKOTNROL. http://www.korg.com/, January 2004.
[5] Clavia DMI. Nord Modular G2. http://www.clavia.se/G2/, March 2003.
[6] JazzMutant. LEMUR. http://www.jazzmutant.com/, January 2005.
[7] Audiopad: A Tag-based Interface for Musical Performance James Patten, Ben Recht, and Hiroshi Ishii. in Proceedings of New Interfacefor Musical Expression (NIME02), May 2002.
[8] B. Bongers. *Physical Interfaces in the Electronic Arts.* Trends in Gestural Control of Music. IRCAM, January 2000.
[9] Microchip. dsPIC. http://www.microchip.com/.
[10] OpenGl.org. http://www.opengl.org/, 2005.
[11] J. Raskin. *The Humane Interface.* Addison-Wesley, March 2000.
[12] dspbox project. http://www.iamas.ac.jp/project/dspbox/, 2005.
[13] Cypress. WirelessUSB. http://www.cypress.com/, 2005.
[14] Bluetooth. http://www.bluetooth.org/, 2005.
[15] Zigbee Alliance. http://www.zigbee.org/, 2005.