VIRTUAL CONTROL PANELS

Przemyslaw Prusinkiewicz and Craig Knelsen

Department of Computer Science
University of Regina
Regina, Saskatchewan, Canada S4S 0A2

ABSTRACT.
A virtual control panel is a collection of controls which are graphically represented on a screen and can be operated according to the direct manipulation paradigm. A control management system makes it possible to interactively create various control panels and interface them with arbitrary applications. In this paper we describe functions which should be supported by an interactive control management system, and present a case study of a system which we have designed and implemented on the Macintosh microcomputer.

RESUME.
Un tableau de commande graphique consiste d'éléments de réglage qui sont représentés sur l'écran d'un ordinateur et peuvent être manipulés d'une manière interactive. Dans cet article nous proposons un système de gestion universel qui permet de créer une variété des tableaux à l'utilisateur. Après avoir présenté les fonctions de ce système d'une façon générale, nous décrivons une réalisation particulière destinée au microordinateur Macintosh.

KEYWORDS: control panel, direct manipulation, MIDI, interface design, message passing.

1. INTRODUCTION

A virtual control panel is a collection of controls (valuators and buttons) which are graphically represented on a screen and can be interactively manipulated using a mouse or a similar device. Thus, it is a metaphor of a physical panel, such as those often found in electronic equipment. According to the traditional approach, the creation, display and updating of control panels were left to applications. This paper introduces the concept of a general-purpose Interactive Control Management System (ICMS) which makes it possible to interactively create various control panels and interface them with arbitrary applications. Thus, an ICMS relieves the application programmer from the burden of interactively acquiring parameters needed to control the application. The impact of an ICMS on the application structure is shown in Figure 1.

Example. Consider a graphics rendering program. The controlled parameters may describe the positions of an observer and a projection plane, current scene illumination, optical properties of the rendered objects, etc. In a simple case, the corresponding panel may consist of virtual slide potentiometers, each controlling one parameter. If the traditional approach to control panels is observed, the code to display and update these potentiometers will be included in the rendering program. In contrast, if an ICMS is used, it will supply the renderer with "ready to use" parameters.

The paper consists of two major parts. In Section 2 we present a "wish list" of features and functions supported by an interactive control management system. It enumerates design issues which must be addressed while developing any ICMS. In Section 4 we present a case study of a control management system called UCofA (a Universal Controller of Anything) which we have designed, implemented, and used for experiments. An outline of related work (Section 3) and a list of problems open for further research (Section 5) are included.

2. FUNCTIONS OF AN INTERACTIVE CONTROL MANAGEMENT SYSTEM

This section further characterizes the notion of an ICMS by describing an extensive set of functions provided by an idealized model of the ICMS. In a particular implementation some of these functions may be absent.

A. Configuration of a virtual control panel.

A.1. Control specification. The ICMS provides the user with a wide range of controls which can be included in a virtual control panel. Some methods of control specification are listed below.

• Selection of controls from a library. This is the simplest and the fastest method of control
Panel definition. In the process of panel definition, the
• Addition of static elements to the windows.
• Group operations. All controls inside a selected
dependent controls, the user indicates its members
and selects the dependency type from a menu.

A.2. Panel definition. In the process of panel definition, the
individual controls are assembled in windows. The
corresponding functions of the ICMS are listed below.
• Window creation. The ICMS provides a mechanism for creating, resizing, repositioning and naming windows. Additionally, superfluous windows can be deleted.
• Control placement. Once an empty window has been created, the user can place controls in it, move them to their target positions, and delete in case of error.
• Group operations. All controls inside a selected window area can be deleted, moved or copied to another area of the same or a different window. This speeds up the process of panel configuration, since a single operation may affect a group of controls.
• Addition of static elements to the windows. Non-manipulable elements such as texts, lines, frames, etc. can be “painted” in the windows for
the purpose of improving the legibility of the panel.

A.3. Control binding. The ICMS communicates with the
application by message passing. The message format is individually specified for each control in a process called control binding. Two binding methods can be distinguished.
• Explicit specification. The message format is specified by an expression in a format definition language. Complex mappings of the control settings to the actual messages (“binding calculations”) are supported.
• Binding by examples. The message format is inferred from examples of messages sent to the ICMS by the application (cf. function B.4). At least two examples are necessary so that the ICMS can identify the constant and the variable fields of the message, as well as the minimum and the maximum value of the controlled parameter.

A.4. Transmission mode specification. The user specifies when a given control should generate a message.
• Triggered transmission. The message is sent once, after the final position of the movable control element (“thumb”) has been attained. Consequently, the transmission is triggered by a specific user action such as the release of a mouse button.
• Continuous transmission. A message is generated whenever the value of the controlled parameter changes. For example, if the parameter value changes continuously from 10 to 25, the messages corresponding to all intermediate values will also be generated. Naturally, the communications channel between the ICMS and the controlled application, as well as the application itself, must support high data transfer rates which may occur in this mode of operation.

A.5. Control exercising. The user can test a panel by selecting the exercising mode. All messages sent by the ICMS are then redirected to a special window on the screen.

A.6. Panel saving and loading. Any panel can be saved to a file and then reloaded to control a given application.

B. Parameter control using virtual panels.
The second group of operations supported by an ICMS is used when an application is actually controlled from a virtual control panel. These operations can be divided into a number of categories.

B.1. Window manipulation. If a panel is large, it may be impossible to simultaneously display all of its windows. A standard window management mechanism is then used to control the screen contents. Windows which are not needed at a given time can be made invisible while the remaining windows can be repositioned according to the user’s preference. If some windows overlap, the active one (containing the control which is actually manipulated) is automatically brought to the front.
panels were incorporated in Infotrol IT, a process-control sys-

purpose utility which could be interfaced with arbitrary

cfgurable control panel was described in [1]. In this case,

objectives of the Panther system of Helfman are particularly

operate a simulated power plant using virtual gauges, valves,

tem designed for use in chemical plants.

values of selected program variables is discussed by Stefik,

was implemented in a simulation system called Steamer

Historically, the first user-configurable control panel

interactive assembling and operating vir-

tual control panels is founded on the general concept of
direct manipulation [15]. The categorization of the ICMS

functions presented in the previous section is influenced by

in that paper, Foley and McMath concentrate on a pro-

cess visualization environment, assuming unilateral flow of

from the process to the viewing system. How-

ever, many notions they introduce also apply to control

management systems. The related problem of using

graphical indicators such as gauges and bars to display

information from the process to the viewing system. How-

ever, Panther does not use

that paradigm.

The first version of UCoFA was presented in [14]. A

refined version is described in the next section.

3. NOTES ON RELATED WORK

The idea of interactively assembling and operating vir-
tual control panels is founded on the general concept of
direct manipulation [15]. The categorization of the ICMS

functions presented in the previous section is influenced by

6. In that paper, Foley and McMath concentrate on a pro-

cess visualization environment, assuming unilateral flow of

information from the process to the viewing system. How-

ever, many notions they introduce also apply to control

management systems. The related problem of using

graphical indicators such as gauges and bars to display

values of selected program variables is discussed by Stefik,

Bobrow and Kahn [16].

Historically, the first user-configurable control panel

was implemented in a simulation system called Steamer

[17]. Steamer made it possible to interactively define and

operate a simulated power plant using virtual gauges, valves,

switches, indicators, etc. However, it was not a general-
purpose utility which could be interfaced with arbitrary

applications. Another example of a special-purpose user-

configurable control panel was described in [1]. In this case,

panels were incorporated in Infotrol II, a process-control sys-

tem designed for use in chemical plants.

Two more recent control management systems were

proposed by Helfman [8], and Fisher and Joy [5]. The

objectives of the Panther system of Helfman are particularly

close to that of our system. However, Panther does not use

the direct manipulation approach to construct control panels;

only the final operation of the ready-to-use panel adheres to

that paradigm.

The first version of UCoFA was presented in [14]. A

refined version is described in the next section.

4. AN EXAMPLE INTERACTIVE

CONTROL MANAGEMENT SYSTEM

Originally, UCoFA was created for controlling param­

eters of musical synthesizers equipped with the Musical

Instrument Digital Interface (MIDI) [11]. In this application,

the need for an external control management system is par-

ticularly evident. On one hand, the number of controllable

parameters provided by modern synthesizers is large — often

in excess of one hundred. On the other hand, the number of

physical controls (knobs, switches and slide potentiometers)
on the synthesizers’ control panels is kept small to minimize

their costs. As a result, users are required to perform

cumbersome routing operations in order to access the desired

parameters. In this situation a potentially unlimited set of

user-configurable virtual panels which can be bound to arbi-

trary synthesizers presents a substantial improvement over the

"real" panels.

UCoFA runs on the Macintosh (*) microcomputer and

makes extensive use of the Macintosh firmware. The user

interface is designed according to the guidelines for

Macintosh. Specifically, notions such as the dialogue win-
dow, GoAway box, button control type, active window and

active control are used consistently with their definitions in

9. The following description outlines the available com-

mands and focuses on two specific aspects of the UCoFA

operation: control manipulation and control binding.

4.1. Menu commands.

Six pull-down menus: File, Edit, Panel, Window,

Control and Communications are available in the menu bar

at the top of the screen in addition to the usual "apple" menu.
The remaining portion of the screen is designated as the

work area where windows and their controls can be

placed (Figure 2).

1. File. Panels are treated by UCoFA as data files. The

File menu contains items for saving and loading these

files, as well as some miscellaneous commands which do

not fit in other menus. In general, the UCoFA File

menu is similar to that found in many commercial

Macintosh applications, for example MacWrite (**).

2. Edit. The Edit menu has four items: Cut, Copy,

Paste and Delete, which are used for text editing.

Copy and Paste are particularly useful while specifying

message formats for a family of related controls.

3. Panel. The items of the Panel menu correspond to the

names of available panels (windows with associated

controls) (Figure 2). Selecting an item brings the panel to

the screen (if not already) and places it in front of all

other panels. A panel which is no longer used can be

hidden by clicking into its GoAway box. The Panel

menu is special in that its items are not predefined: they

are added and deleted following the creation and remo-

val of panels.

4. Window. The Window menu has three items. New

Window creates a new window on which controls can

be placed. Define... makes it possible to name the

resulting panel; this name automatically appears in the

(*) Macintosh is a trademark licensed to Apple Computer, Inc.

(**) MacWrite is a trademark of Apple Computer, Inc.
Panel menu. **Remove** makes it possible to delete a particular panel; its name is then removed from the Panel menu.

5. **Control.** The **Control** menu is used to create controls, specify their attributes, move the controls within a given window and remove them if they are no longer needed. The list of menu items includes all supported control types (variants of vertical and horizontal sliders, a knob, a monostable button, and a bistable button). In addition, the **Control** menu contains the following items:

- **Define.** A dialogue window associated with the current control appears on the screen (Figure 3). By editing the contents of this window the user can define the control’s name, the minimum, maximum and default value of the controlled parameter and the message format associated with the control (c.f. Section 4.3). The user can also specify some attributes affecting the appearance of the control on the screen.

- **Drag.** When a control is created, it appears in a predefined position in the active window. The **Drag** menu item allows the user to move the control to its desired position in the window.

- **Remove.** Removes the active control. The user is asked to confirm this action using a dialogue window.

- **Remove All.** Removes all controls from the active window. Again, the user is asked to confirm this operation.

6. **Communications.** This menu has three items:

- **Setup.** A dialog window appears, allowing for the selection of communication parameters. Both the

RS-232 interface (at various baud rates) and the MIDI standard (31250 baud) are supported.

- **Data windows.** Two special windows, called **Data output** and **Data input,** appear on the screen. The **Data output** window displays the outgoing messages for the purpose of control exercising. The **Data input** window displays messages sent to UCofA. In the context of MIDI applications, this can be used to reveal the format of non-standard ("system exclusive") messages implemented in a particular synthesizer.

---

**Figure 2.** Example of the UCofA screen.

**Figure 3.** A dialogue window used for control definition.
4.2. Control manipulation.

By default, knobs and slide controls are displayed together with numerical fields showing current values of the controlled parameters. The value of a parameter can be changed in two ways:

- By manipulating the movable control element (thumb) using a mouse, or
- By editing the numerical field associated with the control.

The editing of the numerical field is particularly convenient when a specific parameter value must be entered precisely. For example, such a need may arise when controlling frequencies of oscillators in synthesizers.

4.3. Control binding.

Parameters are bound to controls by editing the message definition string associated with each control. This string is displayed in the dialogue window which appears after selecting the Define... item from the Control menu (Figure 3). A simple notation based on format specification used by the `printf()` function in C [10] is devised to specify constant and variable fields of the message. The constant fields can be defined using hexadecimal, decimal, octal or binary numbers. The first character of the message string determines the conversion type actually used. The variable fields are identified by '%' signs followed by optional decimal numbers indicating the fields lengths (the default is one). The definition of the message format is complemented by a specification of binding calculations. This specification has a form of a list of expressions given in reverse Polish notation. Each expression describes the contents of a variable message field as a function of the values returned by one or more virtual controls. The allowable operators are: +, -, *, /, &, |, !, <, >; they denote the four arithmetic operations, bitwise logical operations AND, OR, NOT, and logical shifts in both directions, respectively. The arguments are accessed using identifiers `xn` where `n` is a unique control description number automatically generated by UCofA at the moment of control creation. The active control can also be referred to as `x`. Naturally, the use of constants is also allowed.

**Example.** Consider the following message definition string:

```
H 24 % fe %2, x x1 +, x100 ! 1 <
```

The described message is five bytes long. The first byte is equal to 0010 1000 (hexadecimal 24). The second byte represents the arithmetic sum of the values returned by the current control and control number 1. The third byte is equal to 1111 1110 (hexadecimal fe). The last two bytes constitute a single field which is computed by negating the value returned by control 100 and shifting the result one position to the left.

4.4. Interfacing UCofA with UNIX applications.

Although UCofA was originally designed with musical applications in mind, it can also be used for other purposes; for example to provide data to programs running on other computers. In the simplest case, the application program treats the Macintosh running UCofA as a terminal and periodically reads the incoming information from an input buffer (Figure 5a). Following a different approach, a
components. Unfortunately, numerical methods may be needed to express constraints in terms of geometric constructions instead of unstructured sets of relations. Nevertheless, even if the equations are solved fast enough and the computational overhead related to control manipulation is acceptable, neither constraint-based nor construction-based systems can be incorporated into an ICMS at the present time. First, the process of interactive control definition requires a further simplification. Second, the software for control creation must be integrated with the rest of ICMS - not a negligible task, given that the existing constraint-based and construction-based systems require rather specific programming environments, such as Smalltalk, Cedar or LISP.

ACKNOWLEDGMENT

The MIDI device driver incorporated in UCoF was designed by Rob Morris. Ian Witten and the reviewers brought several important references to our attention. The support from the Natural Sciences and Engineering Research Council of Canada (grant No. A0524) is also gratefully acknowledged.

REFERENCES

[14] P. PRUSINKIEWICZ: Graphics interfaces for MIDI-

5. FURTHER RESEARCH

Some problems related to ICMS design are still open. Two of them are presented below.

5.1. Graphical design of virtual controls.

Graphical forms of controls resembling real knobs, switches, etc. are limited. A departure from the mimetic design brings more freedom into the choice of control forms and can allow for improved data presentation. One possible source of inspiration for the design of unconventional controls is the area of multivariate analysis [2]. Some controls based on the methods for representing multivariate data were presented in [14]. However, a systematic study of control design still remains to be done.

5.2. Interactive definition of control mechanisms.

The problem of interactively defining arbitrary controls is particularly challenging. One approach is to describe them in a special-purpose language [13]. Another possibility is to create new controls using purely graphical methods. According to the methodology developed for constraint-based graphics systems such as Sketchpad [18], ThingLab [4] and Juno [12], controls can be specified by sets of constraints, i.e. geometric relations between control components. Unfortunately, numerical methods may be needed to satisfy these relations when the controls are manipulated. The use of numerical methods is eliminated in construction-based systems such as Gargoyle [3] and L.E.G.O. [7], which separate process polls the serial input and transfers messages to the application using an interprocess communications mechanism (such as sockets in Berkeley UNIX). The application can immediately react to control manipulation and avoid input polling if the interfacing process uses signals to inform it about incoming messages (Figure 5b).

Figure 5. Interfacing UCoF with UNIX applications: (A) with polling; (B) without input polling by the application.


