SDGToolkit: A Toolkit for Rapidly Prototyping Single Display Groupware

Edward Tse and Saul Greenberg
Department of Computer Science
University of Calgary
Calgary, Alberta, Canada T2N 1N4
[tsee, saul]@cpsc.ucalgary.ca

ABSTRACT
Single Display Groupware (SDG) is a field of study that explores how multiple users share a common display. The problem is that it is hard to develop SDG software, for operating systems offer little support (and considerable hurdles) for developing software that manages simultaneous use of multiple input devices, such as multiple mice and keyboards. Yet serious research in SDG demands that we have the ability to rapidly prototype our ideas. In this poster, we present SDGToolkit, a toolkit for rapidly prototyping single display groupware. We describe its features, and illustrate by code how it works.

Keywords
Single display groupware, interface toolkits, CSCW.

INTRODUCTION
While there is much interest in single display groupware (SDG) [1,2,3], these kinds of systems are still notoriously hard to build. Typically, most researchers develop their own specialized applications from the ground up, resulting in SDG that is tedious to build, difficult to modify, and hard to replicate. This problem is exacerbated by our current generation of operating systems that make it difficult to do even the most basic SDG activities. For example, if multiple mice and keyboards are plugged into a computer, a programmer has to do low-level device manipulation to retrieve and process the various inputs as separate streams. Even when this is done, the programmer has to draw multiple cursors (one for each mouse), interpret delta coordinates into window system coordinates, attach a keyboard stream to a window, and so on.

Because we wanted to pursue research in SDG, we decided to build a toolkit that would help us and others rapidly develop SDG interface components and applications. This paper reports our progress on our toolkit, which we call SDGToolkit. In particular, we focus on how we manage input from multiple mice and keyboards. We describe how SDGToolkit appears to the SDG application developer, and then how it works under the covers.

Cite as:
Tse, E. and Greenberg, S. (2002) SDGToolkit: A Toolkit for Rapidly Prototyping Single Display Groupware. Poster in ACM CSCW '2002 Conference on Computer Supported Cooperative Work, November.

Related Work
SDGToolkit is heavily influenced by other systems. MMM (Multi-Device, Multi-User, Multi-Editor) was an early SDG environment that supported input for up to three serial mice [2]; however, this system was only available to the developers. Pebbles explores how personal digital assistants can be used as input devices for SDG [3]. Of course, video game consoles such as Nintendo, Xbox and Sony Playstation manage multiple input devices, but these are not easy to program and are not suitable for productivity software. The closest to our work is MID [1] (Multiple Input Devices), a Java extension that works with Windows 98 to provide support for multiple mice. Unfortunately, MID does not work with later versions of Windows.

SDGToolkit
SDGToolkit provides the basic primitives required to obtain mouse and keyboard input from all standard device types, including both serial and USB. One programs SDG applications in any language compatible with Microsoft .NET e.g., Visual C++, C# and Visual Basic.

To illustrate what a programmer would see, Figure 1 lists the code for a simple SDG application. First, the programmer creates an `SDGcontrol` instance; this handles all attached mice and keyboards. Second, properties of the SDGcontrol are set: for example, the `SetRelativeTo` property instructs the control to return all mice coordinates relative to a given window's upper left corner. The programmer then discovers how many mice and keyboards are attached to the computer (e.g., the `TotalMice` property), and changes the way individual mice cursors appear by indexing each mouse's `picture` and `text` properties. Third, for a tabletop display where people are seated on different sides of the table, the programmer rotates the pointer's text and how the pointer responds to mouse movements through the `TextRotation` and `DegreeRotation` property. Fourth, the programmer takes action on events returned by the various mice and keyboards. As the example shows, the instance returns events whenever any mouse is moved and its buttons pushed, or whenever a key is pressed on any keyboard. Other events also exist, such as `MouseDown` and `MouseUp`. Of particular importance is the first argument `MouseID` or `KeyboardID` which identifies *which* mouse or keyboard

```
Dim WithEvents g As sdgControl

'Create the control & make returned coordinates relative to the main window.
'Also change the way each mouse's cursor will look
Sub Form_Load()
  Set g = New sdgControl
  g.SetRelativeTo Form1.hWnd End Sub
  For I = 0 To g.TotalMice - 1
     g(I).Picture = "cursor" & I * ".ico"
     g(I).Text = "user-" & I
     g(I).TextRotation = I*90
     g(I).DegreeRotation = I*90
  Next I
Exit Sub
'A mouse moved. Show which one moved and its coordinates
Sub g_MouseMove(MouseID As Long, Button As Long,
                        X As Long, Y As Long)
  Msgbox "Moved: " & MouseID & "-" & X & "," & Y
End Sub

'A mouse wheel was scrolled. Show which one scrolled and by how much
Sub g_MouseWheel(MouseID As Long, WheelDelta As Long)
  Msgbox "Wheel: " & MouseID & "-" & WheelDelta
End Sub

'A key was pressed. Show which keyboard and what key
Sub g_KeyDown(KeyboardID As Long, KeyCode As Long, _
              Shift As Long)
  Msgbox "Key: " & KeyboardID & "-" & KeyCode
End Sub


Sub StandardButton_Click ()
  MsgBox "Mouse " & g.LastMouseClick & " Clicked"
End Sub
```

Figure 1. A simple SDG program written in Visual Basic.

generated the event. Finally, each mouse can interact with normal widgets, albeit in a limited way. This is illustrated by the `StandardButton_Click` in Figure 1, a callback to a standard GUI button. Clicking on the button with an SDG mouse also invokes this callback. Because this callback cannot identify what mouse was pressed, the programmer uses the sdgControl's `LastMouseClick` property to find out which mouse it was.

Figure 2 illustrates a basic SDG sketchpad, written in SDGToolkit, designed for a tabletop display. Its program code (not shown, but similar in style to Figure 1) is short and easy to write. We see multiple pointers, one for each mouse. Each pointer comprises customized text and cursor, and is rotated to orient itself correctly relative to the mouse owner. People work simultaneously, and each mouse move generates different actions e.g., drawing differing lines of different colors. A person changes one's drawing color by scrolling the mouse wheel up and down. Each person can also type text on one's own keyboard, where it is drawn near their pointer in the correct orientation.

Internally, we capture mouse events through Microsoft's Raw Input Model (new in XP). Unlike MID [1] which only manages USB mice, we can potentially access *all* system input devices. For example, we added multiple keyboards to an early version of our system with relatively little effort. Raw Input is quite low level: one registers the device it wants to get data from, and then one monitor input activity on the system. SdgToolkit leverages this in several ways. First, it discovers and registers the appropriate devices (mice and keyboards). Second, for mice it takes the delta values and translates these into absolute window or screen coordinates. Third, it implements the multiple cursors corresponding to the various mice by drawing them as low level windows, which are efficient to move and

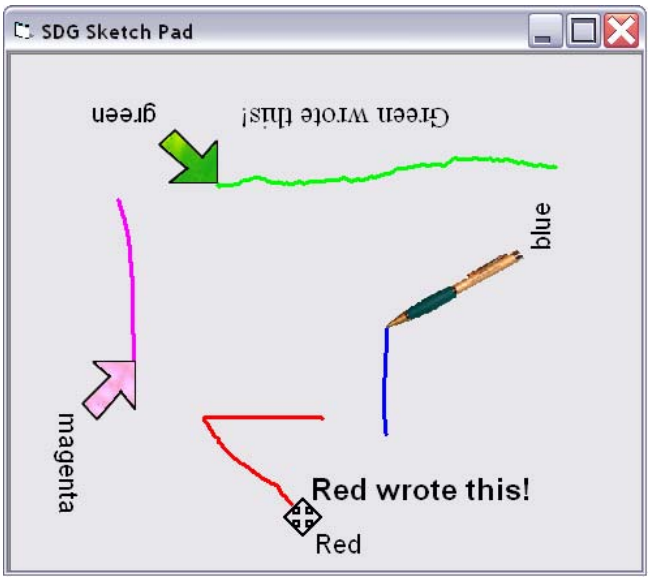

Figure 2: SDG Sketch. Users change colours by moving the mouse wheel up and down

which handle transparency effects. Fourth, it raises events for all keyboard and mice actions, such as those illustrated in Figure 1. Finally, the SDG toolkit invisibly moves the real mouse cursor to the location of the currently active SDG mouse, and raises standard mouse events corresponding to the SDG mouse events. This allows any of the users to interact (albeit in a limited way) with non SDG-aware widgets and standard window controls.

FUTURE WORK

SDGToolkit currently provides only the primitives needed to manage multiple mice, keyboards and pointers. While one can now create SDG applications such as the one in Figure 2, the problem is how widgets (buttons, menus etc) are handled. SDGToolkit does support limited interaction with standard widgets, but it lacks SDG-aware widgets i.e., widgets that respond correctly to different people and/or to simultaneous actions over it. We are currently implementing an extension that lets us and others create true SDG widgets. We will test these widgets, and successful ones will be included in SDGToolkit.

ACKNOWLEDGMENTS. This research was partially funded by NSERC, ASERC, and Microsoft Research. Mike Boyle helped with Windows. Russell Kruger shared his thoughts on tabletop displays.

Software availability: see www.cpsc.ucalgary.ca/grouplab/

1. Bederson, B., and Hourcade, J. (1999) Architecture and implementation of a Java package for Multiple Input Devices (MID). HCIL Technical Report No. 99-08, May. http://www.cs.umd.edu/hcil
2. Bier, E., and Freeman, S. (1991) MMM: A user interface architecture for shared editors on a single screen. *Proc ACM UIST'91*, p79-86.
3. Myers, B., Stiel, H. and Gargiulo, R. (1998) Collaboration using multiple PDAs connected to a PC. *Proc ACM CSCW,* p285-294.