Programming for Multiple Touches and Multiple Users: A Toolkit for the DiamondTouch Hardware

Roberto Arturo Diaz-Marino, Edward Tse, and Saul Greenberg
Department of Computer Science, University of Calgary
Calgary Alberta CANADA T2N 1N4
Tel: 1-403-220-6087
E-mail: saul@cpsc.ucalgary.ca

ABSTRACT
The MERL DiamondTouch is an input device that detects multiple simultaneous touches by multiple people on a surface. While MERL has produced an SDK for this surface, many lines of complicated code must be written to produce even the most basic applications. Consequently, we have created a DiamondTouch extension within our Single Display Groupware Toolkit that considerably simplifies how a programmer captures events from the DiamondTouch. We demonstrate how it works by outlining a multi-user, multi-touch drawing application.

KEYWORDS: input devices, single display groupware.

INTRODUCTION
Many new input devices are now available that considerably expand interface design possibilities. The problem is that, from the programmer's perspective, it is still fairly hard to exploit these devices. The programmer often has to delve into the morass of input drivers, or they have to use a low-level SDK that returns inputs in an unwieldy form. While this makes programming such devices possible for the talented and the motivated, we believe that this extra work inhibits average programmers from rapidly prototyping novel applications. Consequently, one of our research goals is to simplify how input devices present themselves to the programmer. Specifically, we present SDKs and input devices in forms that:

- are in familiar programming languages / environments
- match how people conceptually think about these devices
- make simple tasks achievable with a few lines of code
- minimize housekeeping and other non-essential tasks that have little to do with interacting with these devices.

In this demonstration, we focus on the MERL DiamondTouch Hardware [2] that detects multiple simultaneous touches by multiple people. Each user sits on a receiver (a thin pad). An array of antennas embedded in the surface detects a user's capacitive touch and its location. These touches are reported to a programmer via a basic SDK We have created a toolkit that wraps the DiamondTouch SDK and adds extra capabilities to it, considerably simplifying how people program multi-user / multi-touch applications. This toolkit is embedded within our SDGToolkit [3], a more general toolkit that handles assorted multiple input devices. We demonstrate how this works by highlighting the structure of our toolkit and by outlining a simple drawing program. These and other examples are shown in action in a companion video [1].

Diaz-Marino, R.A., Tse, E, and Greenberg, S. (2003) Programming for Multiple Touches and Multiple Users: A Toolkit for the DiamondTouch Hardware. Demonstration in Companion Proceedings of ACM UIST'03 User Interface Software & Technology.

HIGHLIGHTS OF THE TOOLKIT
The toolkit – available for download – is packaged as a .NET component, making it trivial to include in a standard Windows application. Programmers use several classes provided by our toolkit, briefly described below.

DTManager is the object that encapsulates the capabilities of the DiamondTouch surface and organizes the multi-touch input on a per-user basis. Its properties include:

- `MaximumUsers`: the maximum number of people the DiamondTouch hardware can support.
- `RelativeTo`: a window or control such that events occurring outside of its bounds are ignored.
- `Mode`: the choice of modes give progressively more information about the user's input i.e., whether it emulates a mouse, interprets multiple touches as bounding boxes, or produces x and y vectors containing the raw signal data of a user's multi-touch contacts.
- `TouchThreshold`: the minimum signal strength required from a touch before an event is fired.
- `User`: a collection of `DtUser` objects with information about each user and their recent actions (see below).

The `DTManager` also raises input events, to which people can easily attach callback handlers. All events identify the user that generated them as well as data that defines the event characteristics (i.e., coordinates of touch events). We deliberately modeled these events to resemble mouse down/move/up events, because these are already a familiar programming paradigm. Events include:

- `TouchDown`: when a user first touches the surface.
- `TouchUp`: when a user breaks contact with the surface.
- `TouchMaxMove`: when a the user moves their peak point of contact with the surface.
- `TouchBoxMove`: when the bounding region surrounding a user's multiple touches moves.

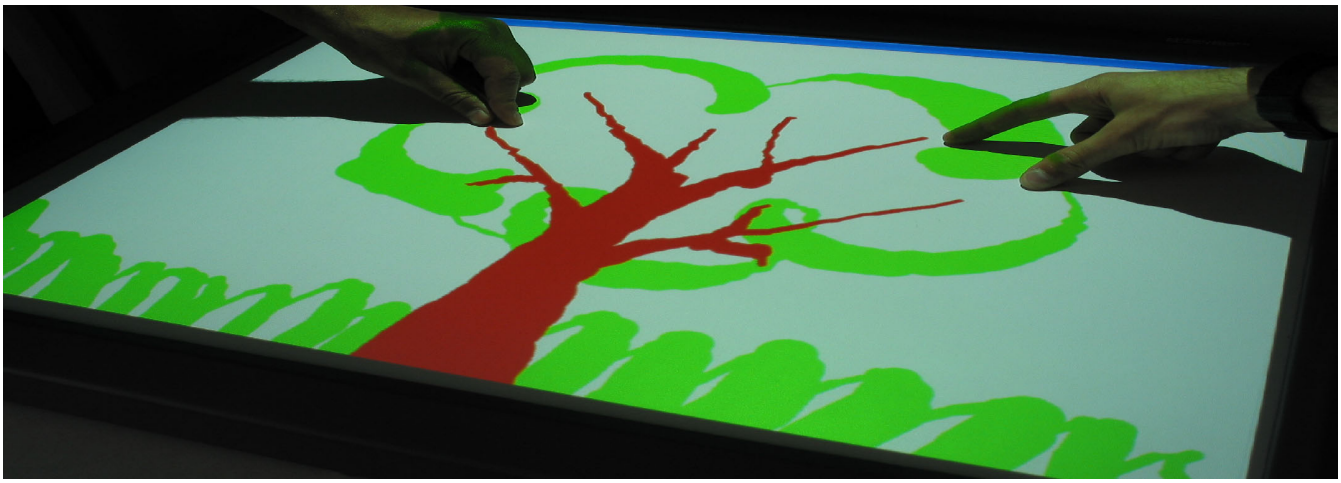


Figure 1: Two people using SquiggleDraw. Line thickness is regulated by bounding box surrounding a user's multiple touches

- `TouchSignal`: a data object containing signal strength vectors across the x and y axis of the surface.
- `Tap`: when a user taps the surface.
- `DoubleTap`: when a user double-taps the surface.

DTUser is an object containing properties associated with each user of the DiamondTouch surface. The DTManager instantiates and maintains these objects automatically in its User collection property. Properties include

- `ID`: a unique number associated with that user.
- `Color`: a unique color to associate with this user.
- `LastBox / LastEvent / LastMax / LastTap / LastSignal`: data about the last event of a particular type generated by the user.
- `Touching`: true if the user is currently touching the surface.

DEMONSTRATION: SQUIGGLEDRAW

We will use SquiggleDraw, a multi-user/multi-touch sketching application, to demonstrate the simplicity of programming with our toolkit (Figure 1). SquiggleDraw has two interesting aspects.

- A person adjusts line thickness on the fly. One draws by changing the bounding region of the drawing with two fingers. One draws thin lines by holding their thumb and forefinger close together, and progressively thicker lines by spreading their fingers apart.
- Up to four people can draw simultaneously, with each person's lines appearing in a different color.

We created SquiggleDraw from scratch in about ten minutes through the following steps. First, using the Visual Studio .Net form designer, the programmer drops a `dtManager` component object onto the window. Second, the programmer registers a callback through form filling that responds to the `dtManager`'s `touchBoxMove` event. As mentioned, this gives the id of the particular user who did the action, and a bounding box that surrounds their multiple touches. Third, the programmer implements the `TouchBoxMove` callback (11 lines of code). This callback draws a line segment from the center of the previous bounding box (obtained by looking up the `LastBox` property `in` the appropriate `DTUser` object) to the center of the current bounding box, in the color of the current user (also obtained from the `DTUser` object). The line thickness is calculated proportional to the bounding box size so that fairly coarse user actions are handled smoothly. Third, the programmer adds a callback to the `doubleTap` event to clear the drawing (one line). Finally, the programmer adds a few other lines to initialize variables, set line appearance properties, and so on. In contrast, we estimate a knowledgeable person building SquiggleDraw with the MERL SDK would require 15-20x this effort as measured by programming time and lines of code.

CONCLUSION

This paper only touches upon the capabilities of the toolkit and the kinds of applications people can build with it. We have created more complex applications; the companion video illustrates a picture-matching game. People tap cards to turn the image over, and use gestures (such as finger-brushing) to enlarge and shrink the photo.

We are now adding graphical feedback. The programmer can request that cursors appear when a person touches the table, or that a bounding box is drawn around one person's multiple touch area. We are also extending the toolkit to include Smart Technology's DViT multi-touch surface.

Toolkit availability: www.cs.ucalgary.ca/grouplab/software/

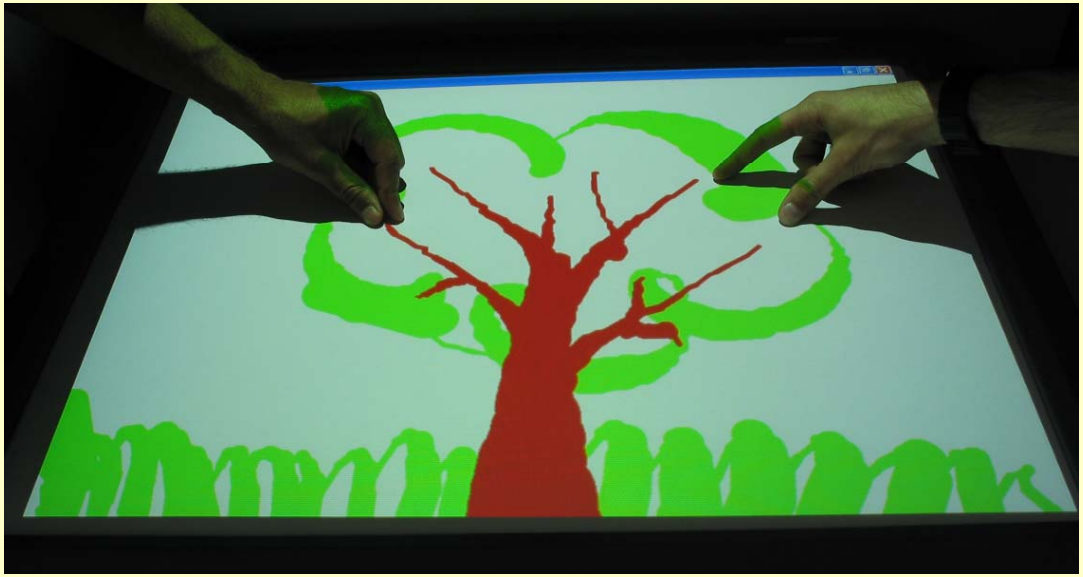
Acknowledgements: Thanks to Kathy Ryall and Joe Marks at MERL for donating the DiamondTouch.

REFERENCES

1. Diaz-Marino, R., Tse, E. and Greenberg, S. DiamondTouch Toolkit: The Video. June. www.cs.ucalgary.ca/grouplab/papers/2003.html, 2003.
2. Dietz, P.H., Leigh, D.L., DiamondTouch: A Multi-User Touch Technology. ACM UIST. 219-226, 2001.
3. Tse, E. and Greenberg, S. Rapidly Prototyping Single Display Groupware through the SDGToolkit. Report 2003-721-24, University of Calgary, Canada. 2003.

Grouplab DiamondTouch™ Toolkit

GroupLab DiamondTouch™ Toolkit simplifies programming MERL's DiamondTouch Tabletop Device.



SquiggleDraw demonstrates how a multi-user multi-touch program can be written in just 15 lines of code.

Only 4 lines deal directly with the input from the DiamondTouch Device (highlighted in bold). The rest is standard drawing graphics

```
private void dtManager_TouchBoxMove( TouchEventArgs e )
{
   Graphics g = this.CreateGraphics();
   if (e.User.LastDown.Timestamp < e.User.LastBox.Timestamp) {

     //Create a pen
      Pen pen = new Pen(Color.Black, 1);
      pen.StartCap = LineCap.Round;
      pen.EndCap =   LineCap.Round;
      pen.Color =       e.User.Color;

     //The last and current bounding boxes
      Rectangle lastbox = e.User.LastBox.Box;
      Rectangle newbox = e.Box;

     // Line thickness depends on the width of the bounding box
        pen.Width = (newbox.Width + newbox.Height) / 10;
        if (pen.Width < 2) pen.Width = 2;

     // Draw a line at the appropriate thickness from the last point to the current point
      Point lastpt = new Point(lastbox.X + (lastbox.Width / 2), lastbox.Y + (lastbox.Height / 2));
      Point newpt = new Point(newbox.X + (newbox.Width / 2), newbox.Y + (newbox.Height / 2));
        g.DrawLine(pen, lastpt, newpt);
     }
}
```

Download the toolkit at: grouplab.cpsc.ucalgary.ca/

This poster accompanied the demonstration.