

Open Letter to OS Designers from the Tabletop Research Community

Dr. Chia Shen

Clifton Forlines

Daniel Wigdor

Mitsubishi Electric Research Labs, Cambridge, Massachusetts, USA (MERL)

Prof. Frédéric Vernier

University of Paris IX, Orsay, France

April 24, 2007

In this letter, we outline three key areas where development is needed to enable a tabletop-friendly operating-system. This list, and its descriptions, has been compiled in consultation with researchers at corporate labs and universities in both North America and Europe.

The *DiamondSpace* project group at Mitsubishi Electric Research Labs (<http://diamondspace.merl.com/>) has been conducting research in to the design and use of interactive tabletop systems since 2001. Among several research products, one offshoot project is the *DiamondSpin* toolkit, now maintained as an open-source project at the University of Paris (<http://diamondspin.free.fr/>). This toolkit is aimed at providing a generalized infrastructure for the production of tabletop applications. A limiting factor in the success of this toolkit has been the need to re-implement many operating system and graphics features, in effect requiring the parallel operation of two such systems: the primary computer operating system, and the tabletop management software. Because of this reimplementations, application designers are forced to produce two versions of their software: one for a regular desktop, and one built with the DiamondSpin toolkit. Because of this, adoption of the toolkit, and tabletops in general, requires significant buy-in from designers.

Far more preferable to such a system would be a tighter integration with the operating system. Such integration would allow for far better performance, and could enable existing applications to run on the tabletop, without the need for reimplementations. In effect, allowing applications to be agnostic to the desktop/tabletop shift. This integration would require significant buy-in from an OS producer.

Integrating Tabletops in to the Operating System

To enable the use of OS on a multi-user tabletop, three fundamental mechanisms need to be developed. We present this list in the order we imagine to be the simplest, to most complex, to implement. This ordering, however, does not reflect each item's priority in terms of defining user experience of the tabletop: each describes an area critical to the successful migration of existing operating systems from the desktop to the tabletop paradigm. These three items are:

- A. Tabletop-friendly controls
- B. Tabletop-friendly graphics
- C. Event architecture for multi user and multi-touch input

Below, we discuss each in detail, along with our ideas for what would be needed to implement these goals.

Goal A: Tabletop-Friendly Widgets

Direct-touch interfaces have two fundamental characteristics: (1) targets can be occluded by the fingers, and (2) the large touch-area of a fingertip needs to be mapped to a screen-point. System-defined UI-controls, such as window manager decorations (min/max), sliders, buttons, menus, etc., need to be reengineered to allow for direct-touch input. Changes include the need to make these controls larger, to compensate for imprecise input, and reposition them to minimize occlusion of window content while in use.

Goal B: Tabletop-Friendly Graphics

Because tabletop systems can be used by multiple users seated at different sides of the table, tabletop-friendly windows need to be painted at arbitrary orientations.



Figure 1. Windows on a tabletop may be rotated to face a particular user.

To enable this, the system would require all of:

1. A mechanism to paint windows at arbitrary positions/orientations
2. A mechanism to have pop-ups, menus, and other such content follow window position/orientation
3. Updating of the graphics engine to correctly render oriented content – especially sensitive content, such as text.
4. Update graphics for resolution independence, since physical size of on-screen graphics is more important with direct-touch interfaces.

Although 2 follows logically from 1, it adds a layer of complexity, since window “content” can sometimes extend beyond the boundaries of its enclosing window:

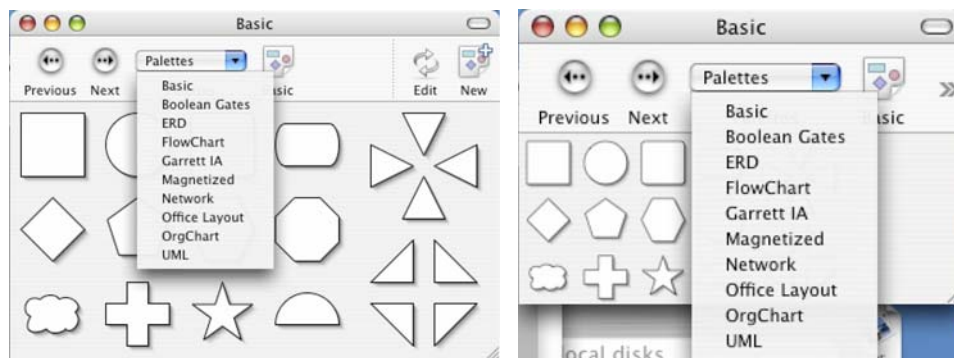


Figure 2. Left: a drop-down menu. Right: the drop-down menu extends beyond the bottom of its enclosing window.

There are multiple schemes for allowing for the intuitive, dynamic reorientation of windows by users of a direct touch interface, described in detail in published works. Which of these schemes is appropriate on a system-wide level is not yet known. Enabling the mechanisms described above, however, would allow for testing of each of these schemes.

Goal C: Event Architecture

When working on a tabletop, there are many fundamental differences in the user input paradigm that challenge core WIMP assumptions:

1. The input area of any given user/point is larger, and a different shape, than the single pixel defined by a mouse pointer.
2. Windows/widgets may be rendered off-axis.
3. A single user may be touching multiple UI-controls/screen points simultaneously.
4. Multiple users may be touching the same UI-controls/screen points simultaneously.
5. Multiple users may be touching different UI-controls/screen points simultaneously.
6. Multiple users may be entering text from different keyboards (soft or hardware) simultaneously.

In order to properly enable a system for multi-user tabletops, each of these changes would need to be made to event delivery mechanisms:

1. Events need additional fields, such as 'touch area shape', 'touch area location', 'touch areas', 'user ID', 'device ID', 'touch pressure/strength'.
2. Focus mechanisms need to be redefined to recognize both multiple input areas by a user, and simultaneous input by multiple users
3. Widget behaviour needs to be defined to properly react to input from multiple users
4. Widget behaviour needs to be defined to properly react to a single user making simultaneous input to multiple points.
5. Multiple keyboard support, be they soft or hardware keyboards, needs to be integrated.

Summary

Enabling each of the mechanisms described above is a first step towards the migration of existing WIMP systems to a tabletop platform. No one issue is a silver bullet – each requires careful planning and execution. Nor will implementing each of these mechanisms immediately create a table-enabled operating system: even after such a system is built, many unanswered research questions remain. The adoption of these mechanisms by an operating system builder, however, would enable a giant leap forward in the development of a tabletop computer.