THE IMI SDK: COMPARATIVE ANALYSIS

PART I: THE ISSUES

An *independent multiple input human-computer interface design* holds obvious and attractive advantages for computer application designers and end users alike. Various approaches to create fully functional and integrated independent, multiple input environments have been attempted with varying levels of success.

An effective solution to independent, multiple human-input device interfacing should encompass at least the following capabilities:

- **Interoperability** – Software should seamlessly interact with widely available devices
- **Device enumeration and configuration** – Ability to easily retrieve and select from a list of available input devices which are to be used by the application
- **Input handling** – Retrieving input data from the selected device(s) and interpreting it in intuitive ways that can be easily interfaced with an application’s input handler
- **Cursor tracking** – Assimilation of device relative and absolute positional data
- **Cursor rendering** – Generation of a visual indicator of the position of the input device
- **User Assisting** – Features must compensate for user uncertainties with non-dominant hand implementations
- **Rapid application integration** – Ability for developers to quickly and easily add independent, multiple input device potential to both existing and new applications
- **Multi-user support** – Allow multiple users to access a single application or multiple applications on a single computer with independent device implementations

Regarding Custom Hardware/Driver-Based Solutions

Development of custom, mouse-like input devices and device drivers is resource intensive in both time and money. The result is a high cost to the end user, and a product that is often only capable of functioning with an application specifically designed to make use of that device. This solution is not interoperable in any respect and other capabilities are dependent on what the specific devices’ software libraries include.

Regarding Generic Driver-Based Solutions

Development of a driver that provides a layer of independent-multiple-input-device functionality is costly, and requires that both the driver be installed and that a (sometimes very specific) version of Windows for which the driver is compatible be installed as well. Furthermore, when three, four and five devices are introduced into a system, other problems of differentiation may arise.

Some attempts at drivers have been made. Problems vary: Offerings of pages of driver install code…without an installer...are available. In some cases, each variation of Windows requires a different code group, and many crashes are reported.

There have been non-productized Linux tests, with Linux operating systems only.

And, above and beyond these limitations, the approaches have not been productized.

Regarding DirectInput

> *“On Microsoft Windows XP, DirectInput enumerates only one mouse and one keyboard device, referred to as the system mouse and the system keyboard. These devices represent the combined output of all mice and keyboards respectively on a system.”* --
> http://msdn.microsoft.com/library/en-us/directx9_c/IDirectInput8__EnumDevices.asp

DirectInput (a component of DirectX) was, prior to Windows XP, an available but incomplete solution for multiple mice.

However, with the introduction of Windows XP this component is no longer available. Therefore, legacy applications that had previously made use of DirectInput to provide independent multiple mouse and keyboard input must be rewritten to make use of alternative methods.

Regarding RawInput

Microsoft now recommends the use of RawInput as an alternative to DirectInput, where multiple mouse or keyboard devices must be treated independently. Unfortunately, the use of RawInput is not a trivial task, requiring much supporting code to be written by the application developer. Furthermore, the RawInput API does not necessarily provide any advantages over DirectInput (the method available prior to Windows XP). Like DirectInput, RawInput does not provide basic functionality necessary in a normal desktop application, such as cursor tracking and rendering which are not required in most gaming applications, nor are any assistive functionalities provided.

Regarding Hardware Solutions

Hardware devices have been developed that incorporate bimanual functionality into their design. Thus far, the issues of cost and design have created barriers in both marketing and usage. The devices require unique chipsets and software drivers, and the associated costs have impacted retail prices dramatically. The design features have specified the non-dominant hand for usage, but the features have often been complex, with many function buttons. This is counter-ergonomic; in the vast majority of interactions, the non-dominant hand plays a supporting role to the dominant hand.

PART II: THE SOLUTIONS

Above and beyond the issues cited, each of the above approaches constitutes a single approach, but none create the architecture for a comprehensive system.

MiceInp, the IMI SDK

IMI has created a patent-pending system interface (MiceInp) that simplifies, enhances and automates the use of multiple mice and other devices, adding new capabilities on top of the core SDK interface. This is the World’s first truly integrated Independent Multiple Input system.

What MiceInp offers is an environment that promotes easy device enumeration, input handling, and cursor tracking and rendering, as well as support for multi-users on a single system. Additionally, MiceInp provides features, such as GRID and OBJECTS that assist users who are unfamiliar with non-dominant hand mouse actions in accurately controlling cursor movements.

MiceInp processes device input data and generates mouse messages that correspond to standard Windows’ mouse messages, making integration intuitive for developers already familiar with Windows’ standard mouse handling messages.

The associated IMI Hardware

IMI has created unique, patent-pending hardware and hardware-design methodologies that require no chipsets or system-level drivers. Costs are therefore dramatically decreased, and the devices associate with the IMI SDK, which provides automation to the non-dominant hand.

Status

IMI is currently developing and implementing new MiceInp SDK designs that call for more and more flexibility and, as the SDK grows, limitations will evaporate and possibilities increase.
Our focus is on the managers who ordinarily have to pay extensive development costs to get the features they want, and who seek unique and differentiating features for their products without the customary associated development time and costs.