NUICursorTools: Cursor behaviors for indirect-pointing
Said Achmiz and Davide Bolchini
School of Informatics and Computing, Indiana University
535 W. Michigan Street
Indianapolis, IN 46202 USA
{sachmiz, dbolchin}@iupui.edu

ABSTRACT
Designing touchless control for six degrees of freedom (6DOF) input devices (e.g., Kinect®) is fundamental for natural user interfaces, but it raises unsolved challenges. These challenges stem from inherent limitations of human motor control in mid-air, and include undesirable jitter from continuous hand tremor, arm fatigue when traversing very large displays, and limited motor control on pixel-accurate selections. To address these problems, we contribute NUICursorTools, a nimble and flexible toolkit that provides a device-agnostic, driver- and middleware-agnostic solution that eliminates touchless cursor jitter, allows optimization of control-display gain and pointer acceleration for touchless control, and enables design of custom complex cursor behaviors. Due to its high-level, device-independent design, NUICursorTools has broad applicability for current and next-generation interaction contexts in which an onscreen cursor is indirectly controlled by any input device.

Categories and Subject Descriptors
H.5.2 [Information Interfaces and Presentation]: User Interfaces---graphical user interfaces (GUI), input devices and strategies.

General Terms
Design, Human Factors

Keywords
Touchless interface, pointer acceleration, control-display gain, jitter, toolkit, C#, NUl, Kinect, Vicon, contours

1. INTRODUCTION
Touchless input with 6DOF (six degrees of freedom) input devices [5] poses a number of unsolved, inherent implementation challenges for visual interface designers. For example, the limitations of human motor control in 3D space cause directional and positional imprecision in hand and limb movement. When a touchless input device is used to control an onscreen cursor, this imprecision manifests in such problems as “jitter” and pointing inaccuracy, especially on very large displays.

Such challenges are encountered whenever an onscreen cursor is indirectly controlled by any of a variety of novel input devices and modalities, such as touchless hand movements via a Kinect®; unanchored device-based movements via a Wiimote® [4]; hand movements tracked by sub-millimeter accurate passive markers, such as Vicon®; or any other nontraditional pointing device used to implement novel NUls (natural user interfaces). In addition to removing jitter and smoothing cursor movement, challenges for interface designers include converting the input data stream from the coordinate system of the control device to the coordinate system of the display device, adjusting control-display gain [1], implementing pointer acceleration, and others. Currently, these challenges must be solved by laboriously writing custom code.

NUICursorTools is designed in a high-level, modular, object-oriented fashion that makes it easy to integrate into an existing code base.

NUICursorTools is based on the following key concepts:
- **Shaper**: A shaper object, which may be instantiated at any point in a project.
- **Transforms**: To configure a shaper, transforms are created, configured, and added to the shaper. Transforms represent specific ways in which input device coordinates must be processed to generate cursor coordinates on the display device, such as jitter reduction or control-display gain adjustment. Creating a shaper and adding transforms each take only 2-3 lines of code (see Figure 1).
- **Coordinates**: Once a shaper has been configured, coordinates may be passed to it as they are received in real time from the input device, and the shaper continuously outputs the processed coordinates of the cursor.

The processed, or “shaped”, coordinates may then be used by the host project to draw the cursor on the display.

3. APPLICATIONS FOR TOUCHLESS INDIRECT-POINTING INTERFACES
Indirect pointing may be used in a laid-back at-a-distance touchless interaction (LATIN) [2] scenario. For example, a pointer on a wall-sized display can be controlled via a markerless motion tracking sensor, such as a Kinect®. Such a setup, in the Advanced Visualization Lab at IUPUI, is used for research meetings and other co-located collaborative work. (See Figure 1 for a realistic artist’s rendering of the space.) We have deployed NUICursorTools at the AVL and are using it for conducting controlled experiments in natural user interfaces.

3.1 Eliminating jitter in touchless cursor
Due to the relatively low resolution (< 30 PPI) of the Kinect® sensor, small, natural hand movements result in a large amount of cursor instability, or “jitter”. Eliminating this jitter is necessary for the interface to take advantage of the full resolution of human limb control. The jitter reduction transform provided
with NUICursorTools, using a dynamic, “smart” smoothing algorithm, stabilizes the cursor with only a few lines of code.

3.2 Optimizing control-display gain and pointer acceleration

Due to the large physical size and resolution of typical wall-sized displays, optimal control-display gain settings are critical in order to allow the user to move the cursor to any point on the display without suffering from motor fatigue, and while maintaining acceptable pointing accuracy. We have used the gain adjustment and acceleration transforms provided with NUICursorTools to address these issues using only a few lines of code.

Pointer acceleration in touchless input scenarios is a largely unexplored area. We are using NUICursorTools as a research platform for experimenting with the effects of various approaches to acceleration on pointing performance.

3.3 Contours for guiding touchless control

We are currently developing a cursor transformation approach for touchless interfaces, called contours. Contours are a complex, context-dependent cursor transformation paradigm that involves constructing a topology underlying the visual user interface and using that topology to dynamically adjust control-display gain. Contours are designed to guide and constrain cursor movements to reflect the user’s intended interface actions. Because of NUICursorTools’ modularity and extensibility, we are implementing contours as an external NUICursorTools module. This also permits contours to be easily combined with any of the cursor transforms that are provided with NUICursorTools.

4. BROADER APPLICABILITY AND CONCLUSIONS

Due to the modular design of NUICursorTools, it is easy for developers to create their own cursor transformation functions and use them alongside the built-in transforms included with the toolkit. Custom transformation functions may replace the functionality of built-in transforms, or may add new functionality. Because NUICursorTools is a high-level library that uses no driver-level or device-dependent code, it may be used with any indirect-pointing input device, ranging from touchless, device-less modalities like the Kinect® or a Vicon® setup, to a multitouch tablet used to control the cursor of a larger display [3]. The device or driver need only supply a real-time stream of 2D point coordinates to the NUICursorTools shaper object.

NUICursorTools is open source software, distributed under the Modified BSD License. It can be downloaded from https://github.com/achmizs/NUICursorTools.

5. ACKNOWLEDGMENTS

We thank the Advanced Visualization Lab at UITS for the use of their facilities.

6. REFERENCES


