Bullseye: An Automotive Touch Interface that's Always on Target

Garrett Weinberg
Nuance Communications
1 Wayside Rd.
Burlington, MA, U.S.A. 01803
{garrett.weinberg,

Andrew Knowles
Nuance Communications
1500 University suite 935
Montreal, QC, Canada H3A 3S7
andrew.knowles,

Patrick Langer
Nuance Communications
Söflinger Str. 100
89077 Ulm, Germany
patrick.langer}@nuance.com

ABSTRACT
We present Bullseye, a novel tactile user interface for carrying out entertainment- and information-related tasks in an eyes-busy environment such as a moving vehicle. Bullseye employs standard resistive or capacitive touchscreens or touchpads, but in a radically simplified form. In a traditional touch-based system, inputs must be carried out at particular XY coordinates corresponding to particular on-screen widgets. Bullseye input gestures, by contrast, may be made on the entire surface of the touchscreen or touchpad without regard to widgets' location. It can thereby enable touch applications that require no visual targeting, an approach that may be preferable to traditional visually-intensive touch applications when considering the constraints of the automotive environment. This paper describes the Bullseye approach and a prototype system built with Bullseye.

Categories and Subject Descriptors
H.5.2 [**User Interfaces**]: Input devices and strategies

General Terms
Design, Human Factors.

Keywords
Touchscreens; touchpads; low-attention interfaces; gestural interfaces

1. INTRODUCTION
Touchscreen interfaces have become ubiquitous since the 2007 introduction of the landmark Apple iPhone. Their fundamental advantage is the combination of information presentation (output) and information manipulation (input) into a single unified area, as opposed to requiring two separate areas, with the screen for output and physical widgets for input. A touchscreen's virtual on-screen widgets can be arranged an infinite number of ways to suit any particular application, and new widgets can be invented to meet new needs.

A prerequisite for the direct tactile manipulation of these on-screen widgets, however, is the ability to see these widgets, and furthermore, to keep these widgets in sight for long enough so that the tip of the finger may be brought into contact with a widget's operable area, a process known as targeting.

In certain operating environments such as a moving vehicle, neither of these prerequisites can be taken for granted. The driver's eyes and hands are occupied with her primary task—safely operating the vehicle—and she may only selectively and briefly attend to secondary tasks (operating windshield wipers, turn signals, etc.) and tertiary tasks (operating in-vehicle information systems, or IVIS).[1] This awareness has been established in human factors and HCI circles for many years. In their 2001 report, for example, [4] Burnett et al. noted that "driver-system interactions should make minimal use of the human visual sense," and they call out touchscreen interfaces as particularly problematic because of their need for targeting and their "basic lack of tactile and kinesthetic feedback."

While great strides have been made recently in bringing tactility to touchscreens (e.g., [11][14][15]), this paper focuses on a simpler, cheaper technique that drastically reduces the visual-motor demand of touchscreens by removing the targeting phase from the touch interaction entirely. After a brief look at the related standards and literature, we will describe our "one big target" technique for touchscreens and touchpads, which we call Bullseye, and explain how we integrated it into an interactive prototype.

2. RELATED WORK
2.1 Standards
Industrial and regulatory organizations have created standards that address the potentially distracting effects of manual-visual interaction with IVIS. The Alliance of Automobile Manufacturers (AAM) Driver Focus-Telematics working group states in the 2006 edition of its guidelines [8] that "[s]ystems with visual displays should be designed such that the driver can complete the desired task with sequential glances that are brief enough not to adversely affect driving." The Japan Automobile Manufacturers Association (JAMA) notes that "drivers must be able to shift their visual attention to the forward field whenever necessary" [12]. The U.S. Department of Transportation incorporated aspects of the AAM and JAMA documents into their recently-issued guidelines to manufacturers on driver distraction [7], and the European Commission has in the past issued similar guidelines [5].

2.2 Research
Many researchers have experimented with touch and gestural interfaces to IVIS. Here we will discuss only a handful of these works that strike us as having the most in common with the Bullseye system.

AutomotiveUI'12, October 17–19, Portsmouth, NH, USA.
Adjunct Proceedings.

[1] In some formulations, the operation of signals, wipers, etc. is lumped in with steering and acceleration/braking as the driver's primary task, leaving IVIS operation to be labeled as the secondary task. See [13] for more information.

Alpern and Minardo conducted a driving simulation experiment using gestures for map- and entertainment-related IVIS control [1]. Their simplification of the gestural vocabulary into directional gestures (up, down, right, left) and numbers (1 – 5) mirrors Bullseye's use of only the cardinal directions for its swipes.

In their pieTouch prototype, Ecker et al. adapt pie menus to the automotive touchscreen context [9]. One adaptation that they make is that the swipes used to select pie "slices" may terminate anywhere on the screen. The same holds true for Bullseye's navigational swipes.

Bach et al. explore using whole-touchpad gestures similar to ours for the control of music playback [3]. Their subjects found the gestural interface "pleasant and less demanding and distracting" than a conventional target-oriented touchscreen and a tactile (pushbutton) interface, and subjects made the fewest lateral control errors and the fewest medium- and long-duration (>0.5s seconds and >2s, respectively) glances away from the road while using the gestural approach.

2.3 Commercial Deployments

While Audi's MMI Touch [2] features an absolute touchpad in its center console in combination with a multifunction rotary knob, we are not aware of any automobile manufacturers that use a targeting-free touchscreen or a relative touchpad in their human-machine interfaces (HMIs).

The closest analog to Bullseye's "whole screen as target" design comes from the smartphone world, in particular the photo browser and Cover Flow music browser [6] that are found on iPhone and iPod Touch devices (most Android phones have a very similar photo browser). We will explain below what distinguishes Bullseye from these implementations.

3. THE BULLSEYE TOUCH INTERFACE

The central idea behind Bullseye is that the entire surface of the touchscreen or touchpad acts as one large input target rather than a collection of various input targets with various predefined active areas.

Swipes in different directions are mapped to different discrete actions. For example, in our current prototype (which will be described in more detail below), a vertical swipe in the downward direction highlights the previous item in a collection of items, whereas a vertical upwards swipe highlights the next item in a collection of items. Unlike conventional swipe-oriented interfaces, with Bullseye only the direction of the swipe matters, not its point of origin, extent or velocity. Likewise only the number and duration of tap gestures matters, not their XY coordinates (direct coordinates in the case of a touchscreen, mapped coordinates in the case of a touchpad).

A single item from one of the item collections is always in focus, and serves as the implicit target of tap inputs.

4. WHAT MAKES BULLSEYE DIFFERENT

4.1 Touch-Somewhere versus Touch-Anywhere

Conventional direct-touch interfaces require the user to target specific points or enclosed areas on the touchscreen, each of which has a given two-dimensional extent. Typical on-screen elements that must be manipulated include virtual buttons, sliders, knobs, etc. Conventional indirect-touch interfaces (often employing touchpads) afford manipulation of a similar set of widgets, typically via a cursor or pointer that serves as a proxy for the user's finger. Indirect-touch interfaces are also typically positional in nature, in the sense that a finger's position on the touchpad is mapped directly to XY coordinates on the screen ([3] is a notable exception in the automotive realm).

Manipulating widgets using either of these forms of touch interface requires a visually-intensive targeting process; users must continually focus on the screen as they guide their finger or their cursor towards the widget of interest.

Bullseye completely removes this visually intensive targeting process, because the user never needs to activate specific areas or widgets on the screen. All operations can be carried out by swipe gestures and taps anywhere on the entire surface of the screen or touchpad. This means that users can operate the application without looking at the screen itself, or with only the briefest of glances to ascertain the result of a swipe or tap. Spoken or non-speech audio feedback may be used to reduce or obviate the need for even these brief glances.

This makes for a more rough-and-ready flavor of touch interaction. Users can vaguely "paw at" the interaction surface while focusing most of their attention on the primary driving task.

4.2 Continuous versus Discrete Swipes

Smartphones, with their limited screen real estate, have necessitated designs that maximize working area by in some cases eschewing conventional widgets (buttons, lists, etc.) and allowing the user to interact directly with the content itself: for example with album art or with photos. In the iOS version of Cover Flow, one can issue the same rough swipe gestures we make use of in Bullseye. However there are two important differences that we feel make Bullseye more suitable for the automotive context:

Firstly, a slow drag or swipe across a given distance on the Cover Flow screen traverses a different number of items than a fast swipe across the same distance. The exact number of items that have been traversed is not able to be deduced without looking at the screen to observe how many pass through the central focal frame. Similarly, a short-distance swipe traverses fewer items than a swipe over a longer distance. In Bullseye, by contrast, both fast and slow swipes over both short distances and longer distances result in the traversal of exactly one item. In other words, our swipe is a discrete navigation operation, incrementing or decrementing a positional counter by one. A Cover Flow swipe, on the other hand, is a continuous operation, moving the positional counter some number of items forward or backward, where that number depends on the velocity and/or extent of the gesture.

4.3 Single-Target, No Exceptions!

The second difference is that, in addition to the swipe gestures, the Cover Flow and photo browser applications feature dedicated areas on the screen that may be tapped to perform certain special actions, such as the 'i' icon in the lower right hand corner of Cover Flow which changes the focal item's display from album art to a track listing. Tapping the album art in focus does the same thing, and tapping another album outside of the focal frame brings that album into focus. A Bullseye application has no such special on-screen targets for tapping; the entire input surface is one big target. Similarly, there are locations on the Cover Flow screen

Figure 1. Bullseye swipe interactions in the Dragon Drive! Demonstrator, illustrated from the perspective of a user in the Music screen (swipe icons courtesy of GestureWorks, www.gestureworks.com).

which, if chosen as the origin of the swipe, cause the swipe to have no effect, for example the status bar at the top of the screen. Our system has no such "dead zones;" the entire surface can be employed for any gesture at any time.

5. PROTOTYPE

In recent months we have built an interactive prototype that employs Bullseye as its tactile interface. Dragon Drive! Demonstrator (DDD) is the reference implementation for Nuance's recently-announced Dragon Drive! Platform. It is a multimodal (voice + touch) content search application whose GUI runs on WebKit-enabled mobile browsers and on the Google Chrome browser for Windows PCs.

5.1 Indexed Navigation Interactions

As explained above, swipes in Bullseye are discrete, relative events rather than continuous, absolute inputs. As such, DDD features an index-oriented navigational paradigm. Horizontal swipes cause the various content domains to slide into view, occupying the whole screen. DDD domains include Directions, Music, News, Phone, Messages and Settings, the latter three of which have only sample content in this prototype (although they are supported by the underlying Dragon Drive! Platform). Within each of these domains is a flat (i.e., non-hierarchical) list of items, the content of which depends on the current search term(s)—or lack thereof—in the given domain.

For example, in the screenshot composite shown in Figure 1, at center we see the Music domain after the user has searched by voice for Art Blakey tracks. If the user swiped from left to right, she would activate the Directions domain as depicted at left. If the user instead swiped from right to left while the Music domain was showing, she would be taken to the News domain. In that domain, if no search taken place yet she sees the most recent headlines from her news feed (we have a content licensing agreement with Time Warner's Entertainment Weekly/EW.com).

Within a given domain, the user swipes vertically to move forward and backward within the current search filter's result list, swapping items into and out of the central focal pane, one item per swipe.

Text-to-speech (TTS) based auditory feedback indicates to the user that the system has processed a given navigational swipe input. After horizontal swipes, the name of the newly active domain is played along with any currently active filter. After vertically swiping to activate the next or previous item in a collection, the new item's title is read aloud by the TTS synthesizer. If a user is already at the top or bottom of the list and tries an invalid vertical swipe, the system plays a "bonk" sound effect to indicate that item traversal is not possible. These forms of auditory feedback are essential for eyes-free operation of the system while driving.

5.2 Contextual Actions and Voice Input

The visual prominence and lighter color of the central pane differentiates the selected item. This item serves as the implicit target of a tap (as opposed to swipe) input. A single tap anywhere on the screen or touchpad carries out the default contextual action upon the focal item. If the item is a point of interest (POI), directions to the POI are shown as a swipe-able turn-by-turn list. If the item is a song, the song is played or paused. If the item is a news headline, TTS playback of the corresponding article is started or paused. A double-tap puts DDD into listening mode, where it can accept both contextual voice commands (e.g., "next paragraph" when listening to news article playback) and global commands/searches (e.g., "directions," "music," "latest news on Broadway," "play Aretha Franklin," "navigate to Starbucks"). A long-tap input (tap and hold for more than one second) clears the current search/filter from the currently active domain, re-populating that domain's list with its default content.

6. DISCUSSION AND FUTURE WORK

As the processing power and storage capacity of IVIS continues to improve and their access to Internet-based content becomes faster and more ubiquitous, system designers face a tremendous challenge in creating safe UIs. In order to keep distraction low and user satisfaction high, they must create intuitive techniques for selecting individual items from a vast universe of possibilities (thousands of POI from an onboard DB, millions of songs from a streaming music service, hundreds of news feeds from hundreds

of Facebook friends, etc.). Many researchers and practitioners feel that search-oriented HMIs might address this challenge better than hierarchical menus with deeply nested functions and interminably long lists of items (see [10] for more discussion of this point).

Bullseye is well suited for such search-centric system designs. In fact without a suitable alphanumeric input solution such as robust ASR (as in DDD) or handwriting recognition (coming soon to DDD), a Bullseye user would have to swipe once per item in the unfiltered list, a completely impractical prospect once the list gets to be over 10 or 20 items in length.

Whether Bullseye's marriage to alphanumeric input technology is acceptable remains to be seen as we flesh out DDD to incorporate more features and thereby more closely resemble a real production IVIS. Certain domains of functionality, such as climate control, clearly don't lend themselves well to list-based representation, calling instead for physical tactile switches separate from a Bullseye-based touchscreen or touchpad. Are there enough of these non-list-friendly functions to make the car dashboard a sea of buttons and knobs just as intimidating and impenetrable as today's screens bristling with submenus and options? Only further, more functionally complete iterations of our prototype will tell us.

It is also extremely important to conduct formal simulator and vehicle-based usability testing of DDD or other systems built with Bullseye. Can we empirically observe the presumed benefits to eyes-on-road time that come from Bullseye's no-targeting-required eyes-free operability? Is the absolute time required for the swipes and taps in a Bullseye search interaction greater or lesser than for a comparable search interaction with a conventional coordinate-oriented touchscreen? If Bullseye interaction times are indeed longer, might this perceived disadvantage be outweighed by better lateral and/or longitudinal vehicle control in the Bullseye case, or by fewer missed stimuli (say, the brake lights of a lead vehicle in a following task)? These are questions only properly designed experiments can answer.

7. ACKNOWLEDGEMENTS

Many thanks to Tim Lynch, Victor Chen, Slawek Jarosz, Rick Megley and Lars König for their help with the refinement of the Bullseye concept and the development of the prototype.

8. REFERENCES

[1] Alpern, M. and Minardo, K. 2003. Developing a car gesture interface for use as a secondary task. In CHI '03 extended abstracts on Human factors in computing systems (CHI EA '03). ACM, New York, NY, USA, 932-933. DOI=10.1145/765891.766078 http://doi.acm.org/10.1145/765891.766078

[2] Audi A8 MMI Touch. http://www.audiusa.com/us/brand/en/models/a8/explore/a8_mmi.html

[3] Bach, K.M., Jaeger, M. G., Skov, M. B., and Thomassen, N. G. 2008. You can touch, but you can't look: interacting with in-vehicle systems. In Proceedings of the twenty-sixth annual SIGCHI conference on Human factors in computing systems (CHI '08). ACM, New York, NY, USA, 1139-1148. DOI=10.1145/1357054.1357233 http://doi.acm.org/10.1145/1357054.1357233

[4] Burnett, G.E., Porter, J.M.: "Ubiquitous computing within cars: designing controls for non-visual use" International Journal of Human-Computer Studies, 55. 2001. pp. 521-531

[5] Commission of the European Communities. 2007. Commission Recommendation on Safe and Efficient In-Vehicle Information and Communication Systems: Update of the European Statement of Principles on Human Machine Interface. http://www.umich.edu/~driving/documents/EU_guidelines_2007.pdf

[6] Cover Flow. http://en.wikipedia.org/wiki/Cover_Flow

[7] Department of Transportation, National Highway Traffic Safety Administration. Docket No. NHTSA-2010-0053. Visual-Manual NHTSA Driver Distraction Guidelines for In-Vehicle Electronic Devices. http://www.nhtsa.gov/staticfiles/rulemaking/pdf/Distraction_NPFG-02162012.pdf

[8] Driver Focus-Telematics Working Group, Alliance of Automobile Manufacturers. 2006. Statement of Principles, Criteria and Verification Procedures on Driver Interaction with Advanced In-Vehicle Information and Communication Systems. http://autoalliance.org/files/DriverFocus.pdf

[9] Ecker, R., Broy, V., Butz, A., and De Luca, A. 2009. pieTouch: a direct touch gesture interface for interacting with in-vehicle information systems. In Proceedings of the 11th International Conference on Human-Computer Interaction with Mobile Devices and Services (MobileHCI '09). ACM, New York, NY, USA, Article 22, 10 pages. DOI=10.1145/1613858.1613887 http://doi.acm.org/10.1145/1613858.1613887

[10] Graf, S., Spiessl, W., Schmidt, A., Winter, A., Rigoll, G. 2008. In-car interaction using search-based user interfaces. In Proceedings of the twenty-sixth annual SIGCHI conference on Human factors in computing systems (CHI '08). ACM, New York, NY, USA, 1685-1688. DOI=10.1145/1357054.1357317 http://doi.acm.org/10.1145/1357054.1357317

[11] Harrison, C. and Hudson, S. E. 2009. Providing Dynamically Changeable Physical Buttons on a Visual Display. In Proceedings of the 27th Annual SIGCHI Conference on Human Factors in Computing Systems (Boston, Massachusetts, USA, April 4 - 9, 2009). CHI '09. ACM, New York, NY. 299-308.

[12] Japan Automobile Manufacturers Association, Inc. Guideline for In-vehicle Display Systems – Version 3.0. http://www.jama.or.jp/safe/guideline/pdf/jama_guideline_v30_en.pdf

[13] Kern, D., Schmidt, A. Design space for driver-based automotive user interfaces. In Proceedings of the 1st International Conference on Automotive User Interfaces and Interactive Vehicular Applications (AutomotiveUI '09). ACM, New York, NY, USA, 3-10. DOI=10.1145/1620509.1620511 http://doi.acm.org/10.1145/1620509.1620511

[14] Senseg electrostatic touchscreen coating. http://senseg.com/technology/senseg-technology

[15] Tactus microfluidic tactile layer for touchscreens. http://www.tactustechnology.com/technology.html