Smartcasting: A Discount 3D Interaction Technique for Public Displays

Krzysztof Pietroszek  
University of Waterloo  
200 University Ave W,  
Waterloo, ON, Canada  
kmpietro@uwaterloo.ca

Anastasia Kuzminykh  
University of Waterloo  
200 University Ave W,  
Waterloo, ON, Canada  
akuzmin@uwaterloo.ca

James R. Wallace  
University of Waterloo  
200 University Ave W,  
Waterloo, ON, Canada  
jwallace@uwaterloo.ca

Edward Lank  
University of Waterloo  
200 University Ave W,  
Waterloo, ON, Canada  
lank@uwaterloo.ca

ABSTRACT
We introduce and formally evaluate smartcasting: a smartphone-based Ray Casting implementation for 3D environments presented on large, public, autostereoscopic displays. By utilizing a smartphone as an input device, smartcasting enables “walk up and use” interaction with large displays, without the need for expensive tracking systems or specialized pointing devices. Through an empirical validation we show that the performance and precision of smartcasting is comparable to a Wiimote-based raycasting implementation, without requiring specialized hardware or high-precision cameras to enable user interaction.

Author Keywords
3D interaction, mobile interaction, 3D displays, 3D environment, interaction technique, raycasting

ACM Classification Keywords
H.5.2 [Information Interfaces and Presentation]: User Interfaces;

INTRODUCTION
Recent advances in display technologies have made cost effective, accessible, and mass deployable large-screen 3D displays possible. For example, autostereoscopic (Lee, Travis, and Lin, 2008; Liao, Dohi, and Nomura, 2011; Travis 1990) and fog displays (Diverdi et al., 2006) enable interaction with large format, 3D content in domains as diverse as medicine (Held and Hui, 2011), construction (Messner and Yerrapathuni, 2003), and natural resource exploration (Bishop and Karadaglis, 1997). The size of these displays, often 1.5m or greater in diameter, naturally supports collaboration around 3D data. It is our belief that the utility of these large displays is maximized when end-users can choose to access and manipulate their content rather than being a passive observer.

To make a 3D display interactive requires an input device capable of performing 3D manipulations. One option to support 3D interaction is specialized hardware in various forms. For example, 3D tracking systems like the Kinect or Vicon computer vision systems provide highly accurate monitoring of an individual end-user's position; users can then use specialized gestures to perform 3D manipulations. However, these systems are frequently limited to two or three simultaneous users. Specialized hardware devices like Nintendo's Wiimote or Thalmic's Myo can be mapped to 3D manipulations and used to track larger number of users, but each user must have the requisite device and carry it with them. Finally, it may be possible to augment a 3D display with direct-touch sensors to support 3D interaction, but this specialized hardware drives up the costs of the 3D display and limits interaction to arm's length.

One question we wish to explore is whether a modern smartphone can be used as an effective 3D input device for large shared 3D displays. Obviously, improvements in technology -- the Kinect, the Myo, etc. -- may improve our ability to track end users with the goal of supporting 3D input. However, we are aware of no research that describes the accuracy of current smartphones as a 3D input device. In the absence of this research, we are left to ask whether smartphones are sufficient to support 3D targeting and translation, or whether improvement in sensors is necessary before effective 3D interaction can be performed on common smartphones. Moreover, without some understanding of the performance of a commercially available smartphone as a 3D input device, we cannot provide guidance to smartphone manufacturers

Figure 1. Through smartcasting, users can interact with 3D content on nearby large displays using their personal mobile device.
on how sensing technology must improve to support 3D input. We seek to provide such a benchmark.

In this paper, we describe an implementation of depth ray with depth marker (Grossman and Balakrishnan, 2006) or depth-cursor, a target-agnostic 3D input technique that uses ray-casting (Liang and Green, 1994) to specify a x, y location on a 3D display and a depth along the ray to uniquely identify a 3D location. We leverage the gyroscope, accelerometer, and touchscreen sensors in a smartphone to control our depth-cursor, and contrast our smartphone performance with the performance of the same technique controlled by another inexpensive, widely available input device: the Nintendo Wiimote. We show that our smartphone implementation performs on-par, statistically, with the Wiimote implementation, and that our smartphone implementation may, in fact, outperform the Wiimote in some experimental conditions. As well, we find no statistically significant change in the cognitive cost of controlling a depth-cursor on a smartphone versus with a Wiimote, despite the fact that the Wiimote is a dedicated ray-casting input device. Together, these results argue for the utility of a modern smartphone as a ubiquitously available input device that can be used to manipulate both 2D and 3D displays (Ballagas et al., 2006; Szalavári and Gervautz, 2008).

The remainder of this paper is organized as follows. First, we present related work in multi-device interaction and in 3D manipulation. We then describe the design of our smartphone-based depth-cursor implementation. Next, we describe and present results of an experiment that contrasts our smartphone implementation of depth-cursor with a Wiimote implementation. Finally, we present a discussion of the smartphone as a 3D input device and contrast alternative design decisions for 3D manipulations with a smartphone.

INTERACTION WITH PUBLIC DISPLAYS

The growing affordability and practicality of deploying large displays into public spaces has motivated the interest of many researchers, as well as industry and the general public (Brynskov et al., 2009). However, while the technological hurdles to deploying these displays have largely been overcome, there remain social, cultural, and other contextual considerations that continue to impede the ability to deploy these large displays at scale (Hachet et al., 2013). For example, research into social behavior around large displays has revealed that many systems never fully engage users because new users are afraid of making “foolish” mistakes and embarrassing themselves in front of others (Brignull and Rogers, 2003; Reeves et al., 2005). This confusion and uncertainty surrounding interaction may increase social resistance to adopt new technology (Huang et al., 2006; Norman, 2002) resulting in deployed technology that is infrequently used (Hornecker, 2008).

Complicating the matter further, many methods of enabling large screen interactions require the use of specialized, expensive technologies. For example, for 3D displays, users may be required to interact through “magic wands” (Cao and Balakrishnan, 2004) or high-precision, high-speed cameras (Vogel and Balakrishnan, 2005). Where lower cost options have become available, such as Microsoft’s Kinect, these still often fail to work well in public environments where distractors such as passers-by can interfere with interaction.

Despite the limitations of technology, techniques have been developed to support 3D interactions with displays. These techniques can be divided into two families: interaction at arm’s length and interaction at a distance. We briefly examine each of these:

Interaction At Arm’s Length

Multi-touch gestures, often similar to the gestures used on mobile devices such as smartphones and tablets, have been used to manipulate objects in 3D environments (Hachet, Bossavit, Cohe, & de la Riviere, 2011; Hachet, De La Riviere, Laviole, Cohé, & Cursan, 2013). These techniques tend to be easy to incorporate into public displays, as a touch sensor can be incorporated into the frame surrounding a large display. However, one obstacle to engaging users with deployed content on public touchscreens is interaction blindness (Huang, Koster, and Borchers, 2008): i.e. users may not realize a screen is interactive. Thus, the interactions may not be discoverable during serendipitous interactions typical of public displays (Azad et al., 2012). Further, the requirement that participants interact directly with a large display may prohibit interaction with content distributed across the entire display, as large public displays may stretch beyond a user’s reach.

Interaction At a Distance

Methods supporting interaction at a distance typically make use of a metaphor such as virtual pointing (Ballagas et al., 2006; Bowman and Hodges, 1997), where an object can be selected when a user intersects an object with a ray originating from an input device, such as a tracker or “magic wand”. Raycasting (Liang and Green, 1994), a virtual pointing implementation often used in practice (e.g. Wiimote (Wingrave et al., 2010), PlayStation Move (Ha, Bai, and Liu, 2011)), has been shown to be easy to learn and use, but requires specialized input hardware (Cao and Balakrishnan, 2004) or body-tracking cameras (Vogel and Balakrishnan, 2005) to implement. This requirement for specialized equipment often prohibits spontaneous interaction with 3D environments deployed in public.

Smartphone Interaction

Today a growing number of end-users have a personal mobile device – a smartphone – that can enable interaction with large, public displays, thereby removing the requirement for a specialized pointing device (Pietroszek and Lank, 2012; Wallace et al., 2008). Ballagas et al. (2006) have previously discussed the advantages of such interactions, but note that “realizing this potential will require intuitive, efficient, and enjoyable interaction techniques for applications in the ubiquitous computing domain”. Our goal is to explore these advantages for interaction with public displays.

SMARTCASTING: ITERATIVE DESIGN & PROTOTYPING

As noted by past research, 3D interaction poses challenges that complicate interaction, including
discovering and selecting objects that are occluded from view (Elmqvist and Tsigas, 2006, 2008; Elmqvist, 2005; Zhai et al., 1996) distinguishing between nearby objects in dense environments (Grossman and Balakrishnan, 2006; Grossman et al., 2004; Steed and Parker, 2004; Wyss et al., 2006), and accurately perceiving an object’s depth (Cipilglu et al., 2010; Cook et al., 2008; Kytö et al., 2013; Liao et al., 2011; Rogers and Graham, 1979).

Furthermore, when located in collaborative environments or public spaces, 3D displays must ideally support walk-up-and-use interactions similar to those that have been proposed for environments containing large 2D shared displays (Shoemaker et al., 2007; Vogel and Balakrishnan, 2005).

Because of their ubiquity and array of sensory inputs, we argue that smartphones represent an ideal computational proxy for serendipitously interacting with 3D displays. Given our focus on interactions via smartphones, our goal was to implement some near-equivalent to direct manipulation for 3D environments using the smartphone as a means for performing the 3D manipulations.

Objects in 3D environments can be positioned and rotated around six degrees-of-freedom. Objects can be translated in space along any of the x, y, or z orthogonal axes to define position within a space mapped via a Cartesian 3D coordinate system. Furthermore, even without translation, objects within a 3D space can be rotated around each of the axes, defining three orthogonal rotations typically referred to as pitch, yaw, and roll. Orthogonal rotations are particularly easy to address using a smartphone as an input device: Accelerometers can provide either an isometric or elastic technique for controlling the rate of rotation, e.g. similar to the way an isometric joystick controls cursor speed on laptop computers so equipped.

On the other hand, less seems known about how well a commercially available smartphone can be used to perform 3D translations.

Of the various 3D translation techniques available to us, the smartphone seems most suited to a ray-casting technique known as a depth ray with depth marker (Grossman and Balakrishnan, 2006), i.e. a depth-cursor. A depth-cursor is exactly analogous to a 3D mouse, where an on-screen indicator (a pointer) serves as a virtual proxy for a user's on-screen location. We, therefore, designed and implemented a depth-cursor technique, where the angle of the ray intersecting the display is controlled via the yaw and pitch of a smartphone and the 'depth' is specified using the touchscreen of the smartphone. We dub this technique smartcasting.

To design an interface that best supports smartcasting, we first describe its implementation. Next, we examine attributes of both the smartphone interactions and the visual feedback provided by the 3D display specifically in relation to occlusion and depth perception. We conclude with a very brief description of the pragmatics of linking smartphones and large displays together and of overloading a smartphone to support both translations and rotations in 3D displays. Note that, while these last two pragmatic points are not a novel contribution, they contribute towards the argument that today’s commercially available smartphones – given sufficient accuracy on input – can support walk-up-and-use interactions with 3D displays.

Designing Smartcasting

As shown in Figure 1, smartcasting uses the pitch and yaw of the phone to perform raycasting on the 2D surface of a 3D display, and uses the touchscreen to control the depth of a cursor along the ray. In this way, any 3D position within the (x, y, z) coordinate system of a 3D scene can be targeted.

When a user holds a smartphone in his or her hand, it frequently moves. As well, users may switch between using a smartphone to control their depth-cursor and using their smartphone to access information. As a result, we need some elegant mechanism to move between the three states typical of input devices: an out-of-range state where the input device is not being tracked; a tracking state where the movement of the input device maps to on-screen cursor movement; and a dragging state where acquired targets are being repositioned on the display. To address this requirement, smartcasting’s behaviour can be characterized using a three state model (Buxton, 1990): it begins in an ‘out of range state’ where its inputs are ignored by the 3D display. After placing a finger on the smartphone’s touch screen, the device shifts into a tracking state where orientation information is relayed as input to the large display. While tracking, a finger up and down (i.e. a ‘reverse’ click) will select an on-screen target, and move interaction into a ‘drag’ state. The full interaction model is depicted in Figure 2.

Ideally, in smartcasting the ray should appear to emanate directly from the smartphone in a straight line from the top of the device. To do this, we must map the yaw and pitch of the smartphone onto the world coordinate system and project out from the user. Smartphones contain a gyroscope and accelerometer. Using the force of gravity, an accelerometer can provide accurate pitch data, and using the gyroscope, a smartphone can sense changes in its yaw angle. However, to accurately measure location on the display and the cursor angle, we must know both the yaw and pitch of the device and a user's distance from the display. Furthermore, gyroscope readings for yaw are subject to drift introducing additional imprecision in the horizontal/x-axis location.

Before embarking on an aggressive design exercise to

![Figure 2. 3-state model for smartcasting based on that of Buxton (Buxton, 1990)](image)
correct for yaw drift and to identify the location of a user via computer vision, we first wanted to explore the severity of the problem. Because the depth-cursor technique uses an on-screen 3D pointer, i.e. a ‘cursor’ to represent a user’s location, it might be the case that, as with a computer mouse, the relative yaw and pitch – rather than the exactly physical yaw and pitch – might be sufficient to allow control of the depth-cursor. In other words, the smartphone might function as a relative, not absolute, position input device (Figure 3).

To test relative versus absolute input, we informally piloted our technique with four graduate students at our institution. We found that, for pitch, there exists a significant tolerance for variations between device pitch and y-axis location on the display. Even a coarse grained location estimated from the device camera can be sufficient to position the initial ray, and relative mappings caused few problems with y-location control. As well, for yaw, initial inaccuracies were insignificant. If one assumes that the phone is pointed at the cursor when movement begins, the yaw angle inaccuracies are easily overlooked during one targeted movement. Yaw angle can be re-set each time a user transitions into state 0, the out-of-range state.

Beyond the (x,y) location of the ray on the 3D display, we also need to control the depth of the cursor using movement of the contact finger on the touchscreen. Two options present themselves: direct mapping of finger position to depth and relative mapping of finger position to depth. In direct mapping of finger to depth, we assume that the y-axis of the smartphone display maps to depth along the ray.

We evaluated each of the two options for finger position to depth mapping. For direct mapping, we found that touch accuracy on a smartphone is limited due to the “fat finger” problem (Siek, Rogers, & Connelly, 2005), making it difficult to acquire small targets for our participants. In contrast, with relative mapping, users needed to clutch (i.e. release and move their finger back on the touchscreen to increase cursor reach) to control the depth of the target. However, based on feedback collected from users, the cost of clutching on a touchscreen is relatively small when compared to the cost of an inability to target. One user noted that our use of the touchscreen was analogous to using a touchpad, where clutching is a frequent, acceptable action to move the cursor longer distances with lower control to display gain.

**Tuning Display-Device Interactions**

**Interacting with Occluded Targets**

On 2D displays, any visible target is available for the user to click on, but in 3D environments targets may be occluded. Occlusion arises when, from the perspective of the user, a target is obstructed by another object or objects. Elmqvist & Tsigas (2008) identify four object interactions that may cause occlusion: proximity, intersection, enclosure, and containment. In cases where a target is partially occluded, most techniques will allow for selection, but will suffer from speed-accuracy tradeoffs resulting from the reduced target size in accordance with Fitts’s law (MacKenzie, 1992).

A common solution to the occlusion problem is to modify the scene by hiding or removing the occluding objects, or by repositioning the viewport so that a target becomes visible. Researchers have also explored interactively distorting the space (Cipiloglu et al., 2010; Forsberg, Herndon, and Zeleznik, 1996) or the viewing projection (Elmqvist and Tsigas, 2006) as well as allowing users to “see through” any occluding objects (Burns and Finkelstein, 2008; Zhai et al., 1994) or prompting the user with haptic, audio or visual feedback (Vanacken et al., 2009). However, these techniques are typically not integrated with selection mechanisms, and few techniques explore the issue of a target being completely occluded from the user’s viewpoint.

To address the occluded target problem, in our implementation the target intersected by the depth-cursor is visible, while all intersected objects that are in front of the target are temporarily removed from view. A target-occluding object is shown again when it becomes the furthest object intersected by the depth-cursor or when the ray no longer intersects it. While a number of occlusion removal mechanisms exist (Elmqvist and Tsigas, 2006, 2008; Elmqvist 2005; Zhai et al., 1996), hiding objects along the ray in front of the cursor seems to be most efficient solution (Vanacken et al., 2007). Moreover, the ability of casual users to easily understand the occlusion removal technique was an important consideration for our design.

**Ray Visibility**

We also considered whether the ray on which the depth marker moves should be displayed. Virtual hand implementations of depth-cursor often do not show the ray (Bowman and Hodges, 1997). Showing the ray could help users identify their depth-cursor, particularly if other depth-cursors are present. On the other hand, rays clutter the display. Given the strengths of both options, we conducted an iterative prototyping exercise. Pilot testing revealed no differences between a displayed or hidden ray for selection time, or for the perceived workload of pilot participants as measured using the NASA Task Load.
Index (Hart and Staveland, 1988). Ultimately, we decided to display the ray during our experimental evaluation.

**Hand Tremor**

We initially considered using a tactile volume button, present on most mobile devices, to perform selection in smartcasting. However during pilot testing, we found that, in pressing and releasing a hardware button, unintentional motion of the device would occur (Bowman et al., 2001). Thus, for small targets, particularly when the user is located far from the display, a cast ray may momentarily move outside of the target, resulting in a “miss”. To address this involuntary motion, selection in smartcasting is performed using a ‘reverse click’, consisting of an up and down motion, on the touchscreen rather than a physical button. Our pilot testing suggested that this choice reduced the impact of tremors, and produced a more accurate selection technique.

**Linking And Overloading**

**Binding Smartphones to Public Displays**

In the context of walk up and use scenarios, it is important to consider the way that a smartphone will quickly and temporarily bind with the public display, preferably ensuring that the connection is secure. Smartcasting connects a smartphone to public displays through a unique URL that is opened in the smartphone’s browser. The user can type in the unique URL or read it from a QR code. The URL can be reached through data connection – there is no need to connect to local wifi network. For further security, the connection between the screen and the phone is relayed through a node.js Websocket HTTPS (SSL) server. Pilot testing revealed that the speed of this server is sufficient for low-latency interaction, and our implementation provided interaction latency lower than 50ms over a 3G cellphone network.

**Overloading Translation and Rotation**

As we noted earlier in this section, 3D manipulation of an object requires six degrees of freedom, three for translation and three for rotation. Our implementation of smartcasting makes use of the smartphone to perform selections and translations via a drag state in our three-state model. To incorporate rotations in a final implementation, we use the volume button on a smartphone to switch mode from translation to rotation while in drag state in the three-state model. When the volume button is not pressed, the smartphone controls our depth-cursor and supports dragging. When a user holds down either the up or down volume button, the smartphone becomes an elastic controller for rotations, using yaw, pitch and roll angles to map change in device rotations to change in angular velocity along the appropriate axes for a selected object.

**EMPIRICAL VALIDATION**

In order to validate smartcasting’s design, we compared the performance of smartcasting using a smartphone against an implementation of smartcasting that used a Wiimote to perform selection tasks in 3D scenes. Our experiment investigated the efficacy of both techniques across different target sizes, and for occluded and non-occluded targets. By comparing the performance of smartcasting with a smartphone against an established hardware device we are able to quantify the degree to which use of smartphone can replace more specialized equipment, and to identify any strengths and weaknesses inherent to smartphone-based 3D manipulations. Our experimental task was heavily influenced by Vanacken et al. (2007).

**Apparatus**

During the course of the study, participants were seated 3m in front of a 55-inch LG HDTV Cinema 3D circularly polarized stereoscopic display that was centered vertically and horizontally in relation to participant’s eye line for best stereoscopic effect. Participants viewed the display through a pair of passive circularly polarized LG glasses. Left and right eye images were provided at 60fps over a side-by-side HDMI 1.4a signal at a refresh rate of 60Hz. All experimental software ran on a locally connected PC with an Intel i7 processor, 16GB RAM, and an NVidia GTX570. A baseline raycasting technique was implemented using a Nintendo Wiimote Plus, which is equipped with a gyroscope and connected over Bluetooth. Our smartcasting technique was implemented on an iPhone 5 that transmitted gyroscope and touch events at 10Hz over a local 802.11n wireless network. The resolution of the touch input was less than 0.07 mm. Our implementation of depth-cursor was identical for both input devices, thus ensuring that the implementation differences do not confound the comparison of results. Figure 4 illustrates the experimental setup and apparatus.

**Participants**

Twelve participants (10 males, 2 females) were recruited on campus from the University of Waterloo to participate in the study. Participants’ ages ranged from 24 to 30 (average = 26.8). Eleven participants were right-handed, one was left-handed, and all participants were screened on a stereoscopic display prior to the study for their ability to order objects by depth. Participants received $10 for their participation in the study.

**Experimental Design**

We used a 2 (INTERACTION TECHNIQUE) X 2 (TARGET SIZE) X 2 (TARGET VISIBILITY) within-subjects design. The study utilized three independent variables: technique, target size, and occlusion. Participants completed trials using each of the smartphone and Wiimote implementations. For the target

![Image](image_url)
sizes, targets with either “small” (0.5º) or “large” (1.0º) sizes provided two levels of index of difficulty based on Fitts’s law (MacKenzie, 1992). Finally, targets were either fully visible or fully occluded upon starting the trial. For the fully visible targets, no distractors occluded or partially occluded the goal target, whereas for the occluded version, the goal targets were hidden by the presence of distractor targets. The order of conditions was counterbalanced using a partial Latin square design.

Our experimental design can be summarized as:

- 2 Techniques: Smartphone or Wiimote
- x 2 Target size: Small or Large
- x 2 Target Occlusion: Occluded or Non-Occluded
- x 14 Repetitions
- x 12 Participants

For a total of 1344 trials.

**Experimental Task**

Participants performed a 3D selection task similar to Vanacken et al. (2007). For each trial, participants first selected a start object in the center of the large display. Once the end of the depth ray entered the start object, the start object would disappear and the destination target, another object on the display, would become selectable. Participants moved the ray to intersect the target. To select the goal target, participants released their finger from the touchscreen (for smartcasting) or Wiimote “A” button (for raycasting). Once the depth ray entered the goal target, the task ended and the screen was reset. If the user accidentally selected a distractor, the distractor’s color changed to light blue to indicate an error and an error was recorded by the system. However, the participant could continue until the goal target was successfully reached.

Each trial scene consisted of a start position, goal target, and 42 distractors (Figure 5). The start object was rendered as a yellow sphere, the destination target as a red sphere, and the distractors as blue spheres. Throughout the experiment the start object had a constant size of 1.0 angular degrees, and was displayed in the center of the 55” display at zero depth along the z-axis. The destination target was placed at a random location on an imaginary sphere with 10-degree angular size radius, with the start target as its center. Thus, the 3D distance between the start and goal targets was constant across all trials, but this distance could include any combination of x, y, and z movements that represented the radius of the sphere.

The positions of the distractor targets were randomly determined, with constraints on their position ensuring that they did not intersect with each other, the start target, or the goal target. The distractor targets were randomly assigned a size between 0.5 and 1.0 angular degrees. The density of distractors was constant in the immediate area surrounding the goal target. Five distractor targets were carefully placed around the goal target forming a cube-shaped voronoi region, as in (Vanacken et al., 2007). The goal target, the entire voronoi region, was further rotated by a random angle for each trial.

For trials where occluded targets were present, a depth ray with depth marker (Grossman and Balakrishnan, 2006) implementation was provided for both the raycasting and smartcasting conditions, and participants were required to reduce or extend the length of the ray to reach the start and goal objects. The depth marker was manipulated by moving a finger on the touchscreen for smartcasting and by the up-down buttons of the Wiimote’s d-pad. If the goal target was occluded, the distractors in front of the depth marker that were intersected by the ray disappeared.

**Procedure**

Participants were first asked to complete a brief demographic questionnaire. Before the experimental trials, each participant was briefed on the smartcasting technique, and screened for the ability to see depth. Then the participants completed 20 training trials: 10 with the Wiimote, and 10 with the smartphone input device. These practice trials allowed participants to familiarize themselves with both techniques, and reduced the influence of learning effects on our analysis. Participants then completed 4 blocks (2 for each input device) of 28 experimental tasks, corresponding to 14 trials for each target size per block. After each block, participants completed a brief post-study questionnaire that examined perceived workload during their trials using the NASA
Task Load Index. In total, each participant’s commitment to the study approximately 30 minutes.

Data Collection and Analysis
All interactions made with the study software were logged to computer files. Selection time was the primary experimental measure, defined as the time taken between entering the start position and reaching the destination target. NASA Task Load Index (TLX) (Hart and Staveland, 1988) data was transcribed into statistical analysis software.

Repeated Measures analysis of variance (RM-ANOVA) tests were conducted to examine differences in selection times between target sizes, target visibility, and depth rendering conditions. Friedman tests were used to examine differences in perceived workload measures between conditions based on NASA TLX data. An alpha-value of .05 was used for all statistical tests.

Performance Results
On average, participants completed each trial in 4.24 seconds (σ = 1.1). Our analysis included all results, including those trials identified as ‘errors’, in which participants initially entered the distractor. We now consider each of our three independent variables: technique, target size, and occlusion.

Technique
Our analysis revealed no difference between the two techniques for target selection time (F1, 11 = 1.68, p = .221, ηp² = .13). Smartphone selections took an average of 4.04s (σ = .77), and Wiimote selections took 4.43s (σ = .585) on average.

Occluded vs. Visible Targets
As expected, selection times for occluded targets were longer than for non-occluded targets (F1, 11 = 83.14, p < .001, ηp² = .884), with small targets (4.75s, σ = 1.19) taking longer to select than large targets (3.72s, σ = 1.05). An interaction effect was also found between target size and occlusion (F1, 11 = 5.61, p = .037, ηp² = .338), where small, occluded targets (6.78s, σ = 1.80) took longer to select than large, occluded targets (5.39s, σ = 1.75, p < .001), however for non-occluded targets our analyses revealed no difference (small targets: 2.77s, σ = .759; large targets: 2.04s, σ = .485; p < .001). No interaction effect was found between technique and target size (F1, 11 = .76, p = .403, ηp² = .064).

![Figure 6. Performance of smartcasting vs. raycasting for a) occluded vs. non-occluded targets, b) small target vs. large targets](image)

![Figure 7. Perceived workload results for Raycasting with Smartcasting, with and without depth-cursor](image)
Perceived Workload Results
We also analyzed participant questionnaire responses for any identifiable trends in their perceived workload across all trials. Our analyses revealed differences in the frustration (p = .002) and mental demand (p = .004) for trials in which occlusion was present. However, NASA TLX data revealed no other differences between the experimental conditions. A complete summary of the NASA-TLX data is presented in Figure 7.

DISCUSSION
Our results indicate that smartcasting with a smartphone offers similar performance to that of our Wiimote implementation. This result validates our goal of supporting efficient interaction without requiring users to carry specialized input devices, and to instead rely only on interactions via a mobile device. Our results also suggest that selections made with smartcasting were faster for occluded targets with a smartphone. When considered as a whole, the effect of technique (smartphone versus Wiimote) accounted for a relatively small portion of the variance in our model ($\eta^2_p = .13$), suggesting that smartphones perform similarly to Wiimotes across different target sizes and degrees of occlusion. Finally, our analysis of perceived workload revealed no differences between techniques.

IMPLICATIONS FOR DESIGN
Our study validates many choices we made while designing smartcasting, and informs the design of cross-device interaction with public displays. We now reflect on these decisions, and on how our findings can inform the use of smartcasting in more general contexts. In particular, we discuss our decisions related to enabling raycasting interactions without real-time/precise location data, and in leveraging a phone’s touch screen to enable more powerful interactions.

Raycasting without Accurate User Tracking
One of the compromises in limiting our design to existing smartphone technology was forgoing the ability to accurately track a user’s position in front of the display, and instead capture only sensor data related to the smartphone’s orientation in 3D space. Our validation of smartcasting demonstrates that such a choice may not be as significant a compromise as many might expect, and that techniques that match the performance and perceived workload of traditional raycasting implementations can be developed without such technology.

Interestingly, this choice also provides insight into how raycasting techniques can support accurate interaction at a distance. By sacrificing a precise measure of the user’s position, and thus the ability to accurately draw an on-screen ‘origin’ of the cast ray, we are also able to improve angular precision for users interacting at a distance. For example, when a user is in close proximity to the screen using a traditional raycasting implementation, they have a stronger degree of angular control over the ray’s position than when standing far away from the display. Smartcasting follows a different model, and is agnostic to the user’s position relative to the large display. This model could be further instrumented to enable more fine-grained control over the mapping between the phone’s angle and the on-screen ray’s trajectory, and provide users at a distance a means of more accurately interacting with on-screen artifacts.

Raycasting with a Touchscreen
We initially explored methods of enabling raycasting using only a smartphone’s gyroscope and physical volume buttons. However, our prototyping process revealed that this choice led to imprecision in terms of selection, as the physical act of pressing a button often interfered with a user’s ability to select small on-screen targets. Consequently, we explored techniques that leveraged the smartphone’s touch screen and developed a 3-state model that allowed for users to disengage from the display, and to select and drag on-screen objects. Our validation then confirmed that the developed technique could facilitate basic interaction with a nearby display.

On the other hand, because our evaluation was intended to verify our design choices, we methodologically chose to constrain our smartcasting design to implement a fair comparison to the Wiimote to minimize potential confounds in our experimental design. Thus, while our technique provides an example of what is possible for smartphone interaction on large displays, there remains a need to more fully explore use the smartphone’s touch screen and additional sensory inputs. For example, it may be beneficial to explore the use of on-screen chording (Davidson and Han, 2006) multi-touch input (Steinicke et al., 2008), or gestures (Vogel and Balakrishnan, 2005) to enable more powerful 3D interactions. Similarly, accelerometers may be used to enable motion gestures (Jeon et al., 2010) through the phone, or to enable interactions through a phone’s rotations orthogonal to the display plane (roll). We believe that in building on the base interactions established in our study, there remains a continuing need to explore serendipitous interactions with public displays, and that the smartphone provides a rich, ubiquitous, and adaptive platform from which these interactions can be enabled.

LIMITATIONS AND FUTURE WORK
Our smartcasting design and empirical validation demonstrates that mobile devices can provide a simple, yet effective means of casually interacting with large displays. However, there remains a need to further validate design choices in situ, and to move the lab-based understanding developed here towards a more general understanding of how such interactions might be used in practice. For example, there remain a number of design considerations surrounding how to best provide feedback to multiple users when displays are used simultaneously by groups in public settings. We anticipate that, given the widespread availability of smartphones and growing affordability of large, public displays, this area of investigation will continue to flourish.

CONCLUSION
We present the design and validation of smartcasting, a novel 3D interaction technique that leverages the ubiquity of smartphones to enable interaction on large, public 3D displays. In addition to describing our design and formal evaluation process, we present implications for the design of large, public 3D displays in general. In particular, we
suggest that the use of mobile, touch-enabled devices, and removing the position information constraints of traditional raycasting techniques provide opportunities to explore new, engaging methods of serendipitously interacting with large displays. Most importantly, smartcasting eliminates the need for specialized input or display hardware, thus addressing a common barrier to serendipitous 3D interaction. In doing so, it provides a first step towards low-cost interaction with 3D displays.

ACKNOWLEDGEMENTS

The authors would like to thank our study participants and colleagues from the HCI Lab at the University of Waterloo. Funding for this research was provided by the Government of Canada (NSERC Discovery Grants Program, NCE-GRAND) and Ontario (ORF-RE Program), and by Google Research (Faculty Fellowship Program).

REFERENCES


Reeves, S., Benford, S., O’Malley, C., and Fraser, M. Designing the Spectator Experience. *In Proc. CHI'05*, ACM (2005), 741–750.


Vogel, D. and Balakrishnan, R. Distant freehand pointing and clicking on very large, high resolution displays. *In Proc. UIST’05*, ACM (2005), 33–42.


