Sphere: Multi-Touch Interactions on a Spherical Display

Hrvoje Benko[1], Andrew D. Wilson[1], and Ravin Balakrishnan[1,2]

[1]Microsoft Research
One Microsoft Way, Redmond, WA, USA
{benko | awilson}@microsoft.com

[2]Department of Computer Science
University of Toronto, Toronto, ON, Canada
ravin@dgp.toronto.edu

ABSTRACT

Sphere is a multi-user, multi-touch-sensitive spherical display in which an infrared camera used for touch sensing shares the same optical path with the projector used for the display. This novel configuration permits: (1) the enclosure of both the projection and the sensing mechanism in the base of the device, and (2) easy 360-degree access for multiple users, with a high degree of interactivity without shadowing or occlusion. In addition to the hardware and software solution, we present a set of multi-touch interaction techniques and interface concepts that facilitate collaborative interactions around Sphere. We designed four spherical application concepts and report on several important observations of collaborative activity from our initial Sphere installation in three high-traffic locations.

ACM Classification: H.5.2 [Information interfaces and presentation]: User Interfaces. – Input devices and strategies; Graphical user interfaces.

General terms: Design**,** Human Factors

Keywords: Spherical display, multi-touch, surface computing, collaboration, single-display groupware.

INTRODUCTION

Spherical displays offer an unobstructed 360° field of view to all users, enabling them to explore different perspectives of the displayed data by physically moving around the display. Viewers can use the spherical nature of the display, their physical body position and orientation, and additional cues from the surrounding environment to aid them in spatially finding and understanding data displayed on the spherical surface. Thus, it is likely that the unique characteristics of the spherical form factor could afford interesting usage scenarios and interaction challenges that go beyond what is possible with prevalent flat displays.

While several commercially available spherical displays exist today [13, 17, 18], such displays are not directly interactive and tend to be used as output-only devices. Any interactivity is usually provided through an auxiliary device such as a trackball or an additional flat touchscreen.

UIST'08, October 19–22, 2008, Monterey, California, USA.

In this paper, we present an implementation of a novel, multi-touch-sensitive, spherical display prototype called *Sphere* (Figure 1). We use Sphere to explore the interactive and collaborative possibilities of spherical interfaces through the development of several concept applications. Our work makes the following three contributions:

First, we outline and discuss the unique benefits of spherical displays in comparison to flat displays. While the challenges of designing applications and interactions are arguably greater for a spherical than for a flat surface, applications can be designed that exploit the unique characteristics of spherical displays to create interesting user experience.

Second, we describe hardware and software components needed to facilitate multi-touch sensing on a spherical display. Sphere uses a commercially available Magic Planet display [13] as its core, augmented by our custom touch-sensing hardware. We also discuss the projections needed to pre-distort data for display on a spherical surface.

Third, we present a set of direct touch interaction techniques – including dragging, scaling, rotating, and flicking of objects – that permit interaction and collaboration around Sphere. We also contribute gestural interactions and user interface concepts that account for the spherical nature of the interface. While general in nature, these interactions were developed within the context of four simple prototype application concepts that help us explore Sphere's interactive capabilities, including a picture and video browser, an omni-directional data viewer, a paint application, and a "pong" style game application.

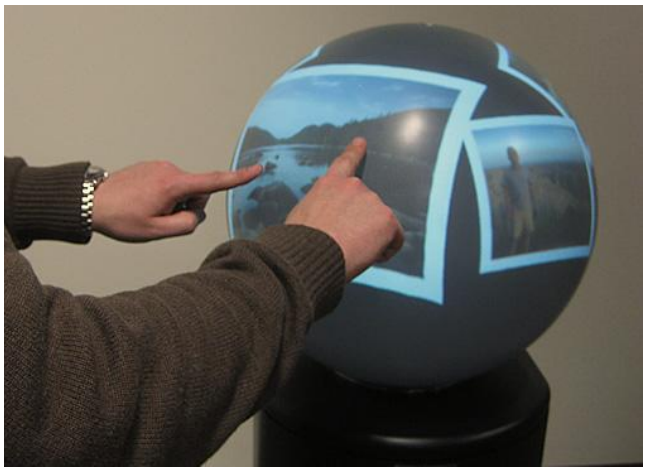

Figure 1: Scaling a picture on Sphere, a multi-user, multi-touch spherical display prototype built on top of a Magic Planet display.

RELATED WORK

Although several research prototypes of interactive spherical and hemispherical displays have been recently presented, none are able to sense and track multiple touch points on their curved surfaces.

Kettner et al. [10] explored interactions with spherical data projected on a spherical surface. Their ViBall display required multiple external projectors and was not directly touch-sensitive, but was able to physically rotate in place, making it behave as a large trackball. This *physical-rotation-only* sensing was well suited for the spherical data Kettner et al. experimented with (e.g., Earth globe images), but did not allow for direct position sensing of multiple touch-points on the spherical surface. A similar physical-rotation-only sensing approach was used in the Globe4D hemispherical display [2]. Unlike the rotation sensing of ViBall and Globe4D, Marchese and Rose [14] used several ultrasonic distance sensors to allow for hand-based interactions away from the display surface. In their system, each sensor independently controlled two rotation axes and one zoom axis.

In contrast to spherical displays that present data on their curved surfaces (a category to which Sphere belongs), spherical volumetric displays have been used to visualize and interact with 3D data within the display. Grossman and colleagues performed several interaction studies on a spherical 3D volumetric display from Actuality Systems, Inc. [7,8]. They found the two most noticeable interaction difficulties resulted from an inability to: (1) display anything on the volumetric display's surface, and (2) physically reach into the display. To alleviate these problems, Grossman et al. created a set of interactions based on modified ray-casting selection from a distance, and used an external motion tracking system to allow gestural interactions with the 3D data.

While incapable of displaying either spherical or volumetric data, the i-ball2 display [24] creates the illusion of presenting data within a transparent sphere. The i-ball2 system can support up to two users, but only by using two independent display systems. Chan et al. [3] designed a system to track hand gestures above i-ball2. They used computer-vision techniques to track the user's hands and pressure sensors at the base of the ball to detect the user's touches. In contrast to our work, Chan et al. do not support spherical interactions or projections on the sphere itself, but rather display data on a regular planar display and interact with it using gestures over a transparent sphere. The glass sphere in i-ball2 is only used to create the illusion of looking at something inside a sphere.

There also exist numerous planetarium-style immersive displays where the user is located within a hemispherical display that is used to create a completely immersive experience. A complete discussion of these displays and various other immersive display technologies is beyond the scope of this paper, but we refer the reader to [1] for further information. Furthermore, while all the spherical displays currently available rely on a projection mechanism for display, in the future, the availability of flexible displays [4] should make various curved displays more common. Even with different implementations of spherical displays, our interaction principles should remain relevant.

From an interaction perspective, our work primarily extends the surface computing research in multi-user and multi-touch interactions (e.g., [5, 9, 15, 19, 25, 26, 27]). Our multi-touch-sensing technique builds on the computer-vision finger-tracking solutions developed by many surface computing prototypes (e.g., HoloWall [15], TouchLight [25], PlayAnywhere [26], and FTIR [9]). In all of these solutions, touch-sensing is performed using infra-red light while projection is done in the visible portion of the light spectrum. This light separation ensures that sensing is not disturbed by the visually visible projected data.

The basic unit of interaction on Sphere is a rotation (quaternion), rather than the translation (vector) common to most flat interactive surfaces. Our interactions are based on Shoemake's work on quaternion-based rotation principles [22] and the ArcBall controller [23].

Research exploring how multiple people collaborate around an interactive tabletop [11, 16, 21] is also highly relevant, as we demonstrate that spherical displays may alleviate some data orientation difficulties commonly associated with tabletop collaboration. Shen et al's. DiamondSpin toolkit [21] enabled arbitrary orientation of all displayed user interface elements to accommodate various user positions around a tabletop. We extend this concept by automatically orienting objects around Sphere to simplify collaboration. We also show how the territoriality concept of Scott et al. [20] applies to spherical displays.

THE DESIGN SPACE OF SPHERICAL DISPLAYS

Most current spherical applications primarily focus on output-only presentations of global data (e.g., weather patterns) or simple marketing applications (e.g., spinning logos or animations). These applications exploit the omni-directional viewing capability of spherical displays, and benefit from their novelty. In contrast, enabling interactivity on spherical displays makes direct manipulation of data and new applications possible. We believe that in order to create compelling interactive applications for spherical displays, it is important to investigate and understand their unique characteristics. While the following analysis focuses on Sphere, it also applies more generally to spherical and hemispherical displays.

Unique Properties

Non-visible Hemisphere: The diffuse nature of the spherical surface makes it impossible for users to see inside the display (unlike true 3D volumetric displays [7]) and ensures that each user, at any given time, can see at most one half (one hemisphere) of the display. While not being able to see the entire display simultaneously may be a disadvantage for some applications, we believe that in many scenarios this presents a unique benefit. For example, multiple people can manipulate data on the same display without disturbing the other users.

No Master User Position or Orientation: In contrast to horizontal tabletop displays for which orientation of displayed content is often a difficult problem [11, 16, 21], spherical displays do not have a "master user" position. In many ways, spherical displays offer an egalitarian user experience, with each viewer around the display possessing an equally compelling perspective. In addition, the orientation of displayed content can be easily adjusted with respect to the prominent physical features of the display, such as the top and bottom poles.

Visible Content Changes with Position and Height: In contrast to flat vertical displays where multiple users share a similar perspective, spherical displays offer each viewer a unique perspective determined by each viewer's position around the display, their height, and the height of the display itself. Even small changes in head position may reveal new content or hide previously visible content.

Smooth Transition Between Vertical and Horizontal Surfaces: A spherical display can be thought of as a continuously varying surface that combines the properties of both vertical and horizontal surfaces. The top of the display can be considered a shared, almost horizontal, "flat" zone, while the sides of the sphere can be thought of as approximating multiple vertical displays. While this is also true of a cuboid or a cylindrical display, spherical displays offer continuously smooth transitions between all such areas. Another way to consider this property is to think about spherical displays as continuously changing in depth and orientation with respect to the user. This also means that for any user's perspective, the best flat surface approximation is the tangential plane at the point closest to the position of the user's eyes.

Pseudo-Privacy: Viewers collaborating around a spherical display have a general sense of which portions of the display are visible to others. Although collaborators are ostensibly free to change position and peek at other portions of the display, such movements are obvious to everyone involved. Consequently, participants can rely on standard social cues to ensure "pseudo privacy" for their actions or content. While spherical displays may not be appropriate for viewing truly confidential data, certain applications, such as games, could exploit this ability to make some actions invisible to others simply by manipulating their location.

Borderless but Finite Display: Spherical displays present a difficult design challenge as they usually require a user interface to be thought of as a continuous surface without borders. With standard flat displays, the content can often stretch beyond the borders of the display, i.e., the display can be thought of as a window into the larger digital world. But for a spherical display, such "off-screen space" usually does not exist; rather, any data moved far enough in one direction will eventually make it full circle around the display. Even when borders are physically present, such as at the base of a spherical display, users tend to mentally perceive this part of the display simply as a hidden portion of a continuous spherical surface.

Natural Orientation Landmarks: Relatively few physical cues exist on the surface of a spherical display. Our informal observations reveal that most people tend to perceive the top ("north pole") as the strongest natural landmark, followed by the equator and the bottom (although the bottom of Sphere is not visible). In addition to these landmarks on the display itself, it is plausible that people can use landmarks in the surrounding environment to help them navigate spherical displays.

SYSTEM HARDWARE AND SOFTWARE

Hardware Implementation

Sphere is based on the Magic Planet display from Global Imagination, Inc [13]. Magic Planet spherical displays use a projector and a wide-angle lens to project imagery from the bottom of the device onto a spherical surface. They are available in a variety of sizes ranging in diameter between 16 inches and 6 feet. The spherical surface of Magic Planet displays is an empty plastic ball coated with a diffuse material that serves as a passive curved projector screen. The bottom of the spherical surface is reserved for the lens and mounting bracket, leaving the displayable portion of the sphere at 290° vertically and 360° horizontally. The quality of the projected image depends on the size of the spherical surface; the brightness, contrast, and resolution of the projector; and the amount of ambient light in the surrounding environment.

Our multi-touch-sensitive Sphere is built on a 34" high podium version of Magic Planet. We experimented with spherical surfaces of 16" and 24" diameter (Figure 2). We use a high-resolution DLP projector (Projection Design F20 sx+, 1400x1050 pixels). Only the central circular portion of the projected image is actually visible on the surface, which effectively reduces the useful resolution to a circle with diameter of 1050 pixels, or approximately 866,000 pixels.

To enable touch-sensing on the spherical surface through the same optical axis as the projection on the surface, we added: an infra-red (IR) sensitive camera, an IR-pass filter for the camera, an IR-cut filter for the projector, an IR illumination ring, and a cold mirror. The physical layout of these components is illustrated in Figure 3.

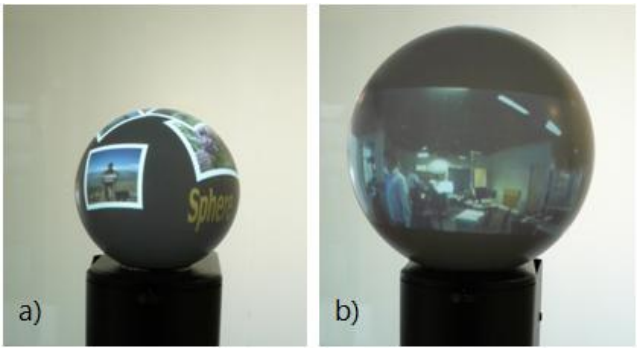

Figure 2: Two sizes of spherical surfaces used in our Sphere prototype: (a) a 16"-diameter ball showing a photo-browsing application, and (b) a 24" -diameter ball showing an omni-directional panoramic video.

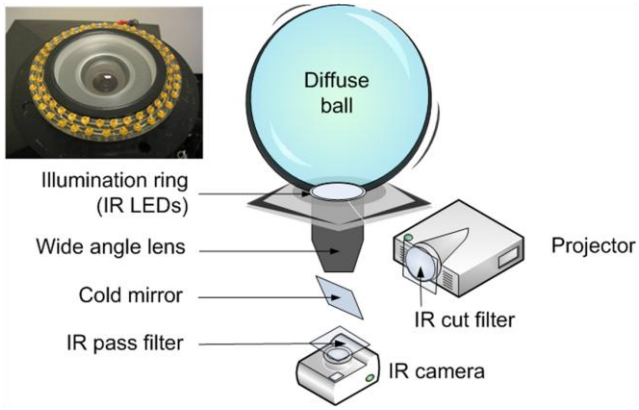

Figure 3: Schematic drawing of Sphere's hardware components that enable multi-touch sensing through the same optical axis as the projection on the spherical surface. The inset picture shows the IR illumination ring consisting of 72 wide-angle LEDs fitted around the wide-angle lens.

Touch-sensing is performed by an IR camera (Firefly MV camera by Point Grey Research with an IR-pass filter) that looks through the same wide-angle lens as the projector. This camera is able to image the entire displayable portion of the spherical surface. To ensure that sensing is not disturbed by currently visible projected data, we perform touch-sensing in the IR portion of the light spectrum, while projection is in the visible spectrum. This approach has previously been used in many camera-based sensing prototypes (e.g., [9, 12, 24, 26]), but not in spherical display applications. We place an IR-cut filter in front of the projector to ensure that the projector emits only visible light which cannot be seen by the IR camera.

The IR light used for sensing comes from a custom ring of 72 wide-angle, IR-light-emitting diodes (LEDs). This ring fits around the wide-angle lens at the base of the sphere (Figure 3). The wavelength of light emitted by the LEDs (880nm) is matched by the IR-pass filter on the camera. Particular care was taken in designing this illumination source to ensure that it provides uniform illumination inside the sphere, but is not directly visible to the camera.

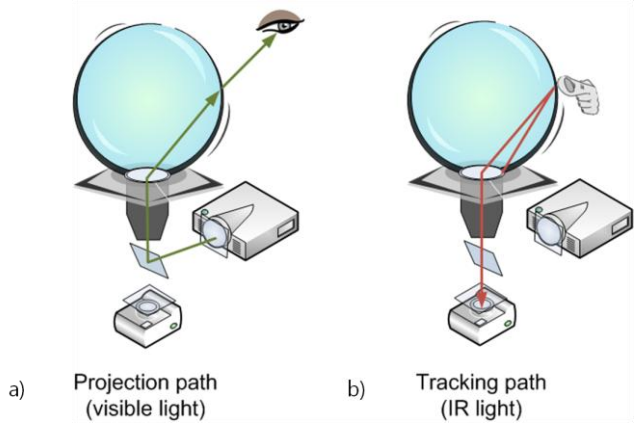

Figure 4: Comparison of optical paths in Sphere taken by: (a) the projection light path (visible), and (b) the tracking light path (IR).

To combine the optical axis of the camera and the projector through a single lens, we use a cold mirror (an optical component that reflects visible light and transmits IR light). Figure 4 shows the difference in the optical paths for projection and sensing. Projected light hits the diffuse surface and is scattered into the eyes of observers; user fingers touching the surface reflect IR light back into the lens to be captured by the camera.

Touch-Sensing Software

In order to track multiple contacts on the surface, the software takes a raw camera image of the entire displayable portion of Sphere, normalizes it, binarizes it, and then finds and tracks connected components in the binarized image (Figure 5).

Any finger or object that touches the surface reflects IR light, and therefore appears brighter than its surroundings in the raw camera image. However, even with careful design, illumination is not completely uniform at different positions on the spherical surface, resulting in contacts on the top of the sphere (the center of the tracked image) appearing significantly brighter than contacts close to the base (as can be seen in Figure 5a). A normalization step accounts for these varying levels of illumination by computing an image where all pixel values are normalized with respect to the minimum and maximum brightness observed at that location. The normalization procedure requires that during initial calibration, we capture a minimum brightness image (i.e., an image of an empty surface), and a maximum brightness image (i.e., an image of a completely covered surface).

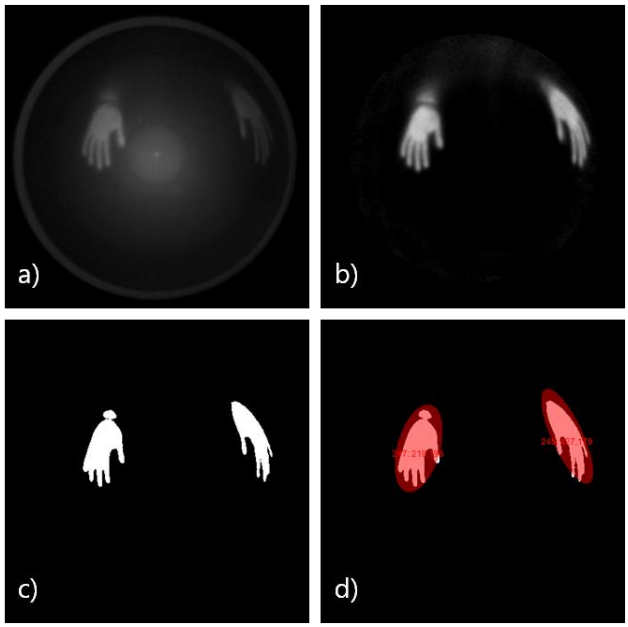

Figure 5: Different stages of our touch-sensing software: (a) raw image of two hands touching Sphere's surface, (b) normalized image, (c) binarized image, and (d) labeled binarized image with two tracked and connected components.

Our tracking software is written as a standalone C++ library and can be used by applications to receive touch updates. The tracking library runs at approximately 30 frames per second with a camera resolution of 640x480 pixels. As for projection, the effective tracking area is constrained to the circle in the image that represents the view through the wide-angle lens a circle of approximately 400 pixels in diameter seen in Figure 5a.

Sensing and Projection Distortions

The wide-angle lens introduces significant distortions that need to be accounted for in both sensing and projection. The sensing camera is imaging a flat radial image (Figure 5) that is subsequently mapped onto a spherical surface to report touch contacts in a 3D Cartesian coordinate system. The projection of data onto the spherical surface requires the use of the inverse mapping, i.e., the data in 3D Cartesian coordinates need to be flattened into a flat radial image for the projector (Figure 6a). This means that displayed objects need to be pre-distorted (flattened) in order to appear undistorted when projected (Figure 6d).

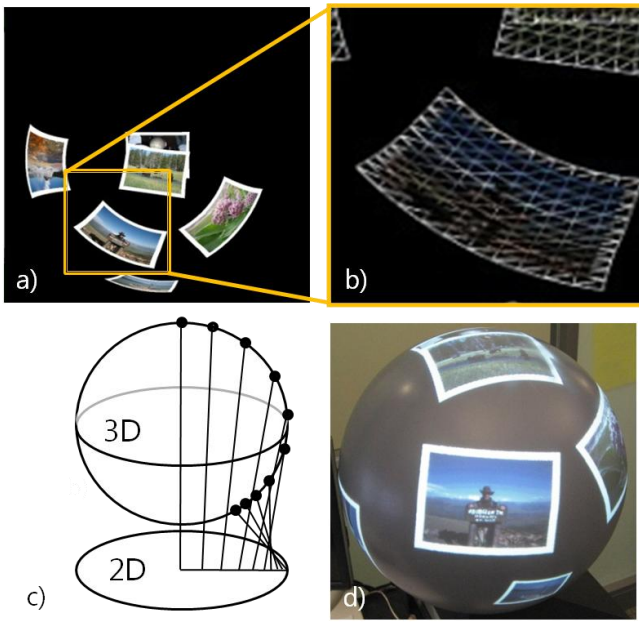

Figure 6: Sphere projection distortions: (a) a radial image displayed by the projector in which all objects are pre-distorted by our vertex shader; (b) a wire-frame view of a portion of the image (a) that reveals high tessellation of objects; (c) a mapping used to create the radial image maps each point on a 3D sphere to a 2D disk; (d) objects rendered on Sphere appear without distortions.

The mapping depends on the physical size of the spherical ball, as the center maps to the top of the sphere and the distance from the center corresponds to the particular height (latitude) on the sphere (Figure 6c). The mapping is determined once during a separate calibration step and can be saved and reused later for all surfaces of the same size.

For non-interactive applications, the distortion of data could be pre-computed off-line and simply replayed when desired. In fact, most existing spherical applications use this approach. However, user interaction with displayed content requires that the system support real-time computation of distortions. To achieve this, we wrote a custom vertex shader to compute the position of each vertex in a radial image at every frame. In our approach, the quality of the distorted image depends greatly on the number of vertices the object possesses; therefore, we highly tessellate each displayed object (Figure 6b).

The distortion requirement makes it impossible to author applications for Sphere using standard graphical user interface toolkits, as these were designed primarily for flat, two-dimensional interfaces. All of our current applications are written in C# using Microsoft's XNA 2.0 framework and use our custom vertex shader to handle distortions. We run these applications on a PC with a 2.67 GHz Intel Core2 processor, and NVIDIA GeForce 8600 GT graphics card.

Data Coordinate Systems

Although it is possible to create content for Sphere in the 2D coordinate system of the projected radial disk image, the distortions described in the previous section make this approach challenging. However, this works well for setting the entire background to a texture in which distortions are not clearly noticeable. For example, the background image of our circular menu (Figure 11) is authored this way.

Alternatively, authoring content in a cylindrical projection is relatively straightforward, as everything is performed in a 2D plane (cylindrical map) which then is mapped onto a sphere. All currently available commercial spherical displays are primarily used for displaying spherical data (e.g., visualizations of planets and stars), and such data is usually stored in a 2D map using an equidistant cylindrical projection (e.g., a flat 2D map of the Earth). However, using cylindrical projections has several well-known distortion problems; these are most visible at the top and bottom of the map (the poles of the sphere). The entire top row (or bottom row) of the cylindrical map is mapped to a single point at the pole. Using such projections makes it difficult to display rectilinear objects near the poles.

Another approach is to author content in 3D Cartesian coordinates in which all objects lie on a unit sphere centered at the origin (Figure 7). Although this approach is more difficult, as it requires all content to be specified in 3D coordinates, it does not suffer from the distortion problems associated with cylindrical projection and offers additional advantages, such as being able take advantage of 3D game engines to incorporate shadows or game physics.

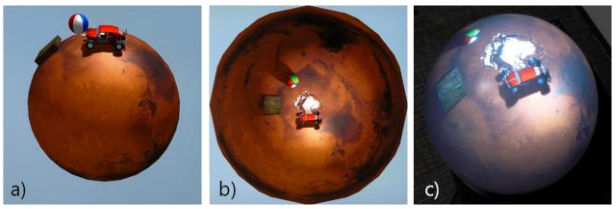

Figure 7: Sphere content authored in 3D Cartesian coordinates: (a) a virtual view of a 3D scene; (b) the same scene flattened to a radial image for projection; (c) the scene when displayed on Sphere.

Finally, a sphere at any given point can be considered locally flat. This assumption allows one to design a relatively small portion of the interface completely in 2D, and to then simply project this flat image from a tangential plane to a point on the 3D spherical surface. This locally flat approach is how photographs are displayed in Figure 1.

Ultimately, the choice of which coordinate system to use for authoring will depend on the content itself and we anticipate that data authored in different coordinate systems will be combined together in the same application.

MULTI-TOUCH INTERACTIONS

We now discuss various multi-touch interaction techniques we developed for Sphere. Enabling user interaction on a spherical surface requires the implementation of basic operations such as selection, translation, rotation, and scaling, as well as providing support for browsing and task switching. While implemented within the context of four simple prototype applications, our interactions are general and designed to be useful to other applications on spherical or cylindrical displays. We also discuss implications of these interactions for multi-user collaboration around Sphere.

Sphere Photo & Video Browser

The first application we explored is a simple browser in which users can freely manipulate rectangular projections of images and videos (Figure 1). Such browsing applications have frequently been demonstrated on flat multi-touch prototypes (e.g., [9, 19]) as they tend to clearly show the capabilities of touch-based, direct manipulation interfaces.

Basic Multi-Touch Object Manipulations

Each object (photo or video) in our Sphere Photo & Video Browser can be independently dragged, rotated, and scaled. As with most touch-sensitive applications, selection of an object is implicitly triggered by any touch contact that lands on that object. Landing on a video object acts as a simple playback toggle: it starts the video playback if the video is paused or stops it if the video is running.

Dragging: Enabling a user to drag an object around Sphere is not as obvious as it might seem at first. The difficulty is that the curved geometry of the spherical surface is drastically different from 2D flat space. In Euclidean space (standard 2D and 3D environments fall into this category), movement is best represented by a displacement vector which encapsulates the direction and magnitude of the movement in a particular line. However, the spherical surface is not a Euclidian space and there are no straight lines on the sphere as all "lines" are actually curves and thus more accurately represented as arcs. While in some cases Euclidean geometry might offer a reasonable local approximation, representing displacement on a sphere with vectors ultimately leads to problematic behaviors.

Instead of vectors, the movement on a sphere is best represented by a quaternion, i.e., a rotation (Figure 8). This has some profound implications for our interactions, as the standard *translation + rotation + scale* manipulation model used in 2D and 3D environments becomes a *compound rotation + scale* manipulation model on Sphere. We stress

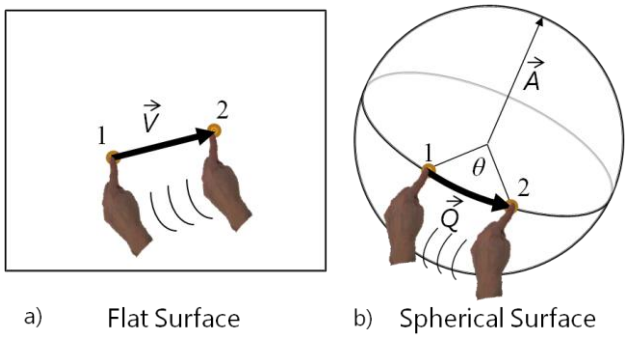

Figure 8: A comparison of basic dragging manipulations when moving a finger between points 1 and 2: (a) on a flat surface this movement is represented by a 2D vector *V*; (b) on a spherical surface this movement follows an arc of an angle θ around an axis *A* (a 3D vector), which is a rotation best described by a 4D quaternion *Q*.

that the rotation is often a compound action as the object is spherically "positioned" by a rotation around the sphere's origin, and then often oriented further in its local coordinate system (see the Local Rotation section below).

We adopted the use of quaternions as Sphere's basic unit of movement, and we model our interactions on rotational principles similar to those discussed by Shoemake [22, 23]. While the system treats all dragging interactions as global rotations around the origin of the sphere, users tend to think of them from their local context and perceive them as a physical translation of an object around the sphere. When asked during demo sessions, users described the interactions purely from their local coordinate system, e.g., "moving to the left" or "this object is above the other one."

Local Rotation: In addition to allowing the user to position (i.e., rotate) an object on Sphere, we facilitate additional 1D adjustment of the object's orientation in its local coordinate system, similar to in-plane rotation of a picture on a flat surface. This operation requires at least two contacts to be touching the object; we map the local angular difference between those two contacts to a 1D rotation of the object. For all our basic manipulations, when multiple contacts are touching a particular object, we aggregate their behavior and apply the aggregate action to the object.

Scaling: Users can scale a picture or a movie by moving their fingers closer together or further apart on top of the displayed object (Figure 1). We map the change in arc length between the touch points to the scaling factor applied to the object. Given that Sphere is a borderless but finite display, scaling an object larger has the potential to envelop the entire surface. While in a single user scenario this might be desired in order to enlarge a particularly small feature, scaling by more than a single hemisphere greatly affects other viewers and ultimately results in either seams or heavy distortions on the other side of the display. In our Sphere Photo & Video browser, the scale of an object is currently restricted to fit within one hemisphere. This seems to be a good compromise between manipulation flexibility and reduced disturbance to other users.

Facilitating Collaboration

We found that dragging an object around Sphere often left it in an awkward orientation (Figure 9a–c). Consequently, most drags need to be followed by a local rotation to correct for the object's orientation. This was particularly obvious with longer drags, e.g., dragging an object over the top of the display to show it to a collaborator results in an "upside down" orientation.

Auto-rotation: To eliminate some of these orientation problems, we enabled each object to automatically orient itself so that the top of the object always pointed to the top of the sphere (Figure 9d–f). If the object was placed exactly on the top, it would remain in its current orientation, until dragged away. This auto-rotation behavior effectively eliminates the need for explicit user controlled local rotations, which is potentially most beneficial in collaborative scenarios. As discussed previously, the top (the "north pole") is a natural orientation landmark that provides an omni-directional "up" cue. Because the system continuously orients all objects with respect to the top, the user is able to simply pass a picture to a collaborator on the other side; the image automatically adjusts its local orientation so that viewing is correct. Surprisingly, this behavior is very much expected; during many Sphere demonstrations, users did not even notice that all objects automatically oriented themselves. Presumably, this behavior appeared natural to viewers because it always resulted in the "correct" view. The auto-rotation feature only became apparent when users attempted to locally rotate an object, an action which was now disabled.

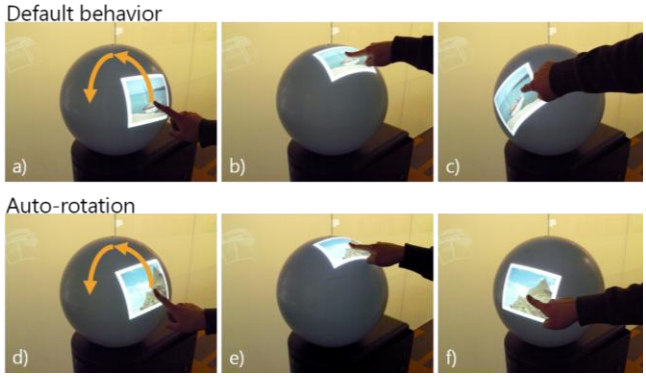

Figure 9: Comparison of orientation difficulties when dragging an object along a similar path shown in (a) and (d): (a–c) default unassisted behavior results in a difficult object orientation visible in (c); (d–f) Auto-rotation behavior eliminates such problems by continuously orienting objects with respect to the top.

Extending User Reach

In many scenarios, it is important to be able to place objects on the other side of the display. This becomes particularly important when collaborating with a viewer standing on the opposite side of Sphere. Although a user can simply drag an object to the other side, this action is tedious if repeated often as it requires extensive physical movement. We implemented two interaction techniques to facilitate this action by further extending the user's reach: *flicking* and *send-to-dark-side*.

Flicking: We added inertia to our manipulations to allow the user to flick an object in a particular direction and have the object continue traveling in that direction without explicit further guidance (such as with a finger). The speed and direction of movement before the user releases an object determine how far and in which direction the object will travel. One can also programmatically vary the angular deceleration (friction) of individual objects in order to ensure that it is easy to flick objects to the other side of the display. By touching an object while in motion, the user can catch it and stop further movement.

Send-to-Dark-Side:*[*] This interaction allows the user to explicitly warp an object and send it instantaneously to the other side of Sphere. To perform send-to-dark-side, the user places a flat palm on top of the object and waits one second. The object is then warped to the mirror position on the other hemisphere. Rather than sending the object directly to the opposite point of Sphere, we decided to simply mirror its position around the plane passing through the top and the bottom of the sphere (shown as dashed line in Figure 10). A significant benefit of send-to-dark-side is that instead of flicking an object and guessing its destination, the user can explicitly control where the object will appear by first manipulating the object's position in its current hemisphere.

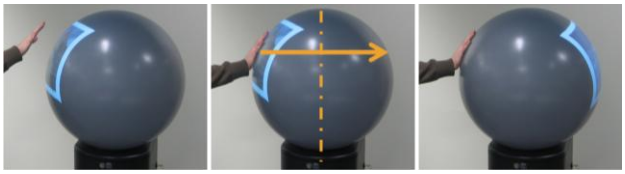

Figure 10: Send-to-Dark-Side: The user touches an object with a flat hand and waits 1 second, resulting in object being warped to the other side of the Sphere.

Both the flicking and send-to-dark-side interactions benefit from enablement of auto-rotation because the latter prevents objects from arriving in the other hemisphere upside down and thus in need of reorientation.

While designed within the Sphere Photo & Video Browser application, the combination of basic multi-touch manipulations and the additional interactions that facilitate collaboration and extend the user's reach form a powerful interaction toolset that can be used for designing other compelling applications for the Sphere.

Sphere's Circular Menu

The abilities to switch between tasks and to select different options are often needed in an interactive system. In Sphere, a circular menu allows the user to select between multiple applications. Currently, the menu is displayed in a circular arrangement around the top of the display and therefore visible to most users; however, if a location of the user were known, it might be better to place the menu in a semi-circle facing the user. Selection is performed not by touching an option but rather by rotating the menu in place

[*] We named this interaction after the Moon's dark side, which is not illuminated by the Sun.

(similar to the JDCAD menu [12]). The highlighted option is selected upon contact removal.

Orb-like Invocation: To invoke a circular menu, the user places two hands (in an open palm posture) on top of the display in a symmetric arrangement (Figure 11). The circular menu fades in accompanied by a sound effect to enhance the experience. Similar to the interaction concept of i-ball2 [3], this gesture is designed to evoke the feeling of interaction with a fortune-telling magic crystal ball. While playful and magic-like, this gesture is highly memorable, easy to repeat, and relatively hard to invoke inadvertently. The size of the two contacts (palm-sized contact is substantially larger than most other touch contacts) and the particular symmetric arrangement of this gesture ensure that the menu is not easily triggered in error.

By combining the orb-like invocation with selection by rotation rather than direct touching, we enabled task switching to occur in one continuous interaction (place hands to invoke, rotate into place, and lift-off to select).

Figure 11: Orb-like Invocation: a bimanual gesture that invokes a circular task-switching menu.

Sphere Omni-Directional Data Viewer

Omni-directional images – such as cylindrical maps of any spherical object or 360° panoramic images – are well suited for display on Sphere. To explore user interaction with such data we designed Sphere Omni-Directional Data Viewer. Examples used were a live-stream from an omni-directional video conferencing camera (Figure 2b), omni-directional images of a city captured by a camera mounted on a car roof (Figure 12a), and the Earth's surface (Figure 12b).

The fact that omni-directional data usually spans the entire display surface presents interesting implications for multi-user, multi-touch collaboration scenarios. Allowing more than one person to touch the data often results in conflict (e.g., multiple people trying to spin the globe in multiple directions at the same time). While restricting interactions to a single touch does mitigate some of the problems (e.g., the first touch assumes control), such a solution is often confusing to the other Sphere users. While this issue should be investigated further, in our current system users are left to socially mitigate such situations: either taking turns or allowing one person to "drive" the interaction

Tether: We allow the user to rotate and inspect omni-directional data; however, this often causes data to be left in an orientation potentially confusing to others. To mitigate this problem, we implemented a tether behavior. Tether allows free manipulation of data via touch contacts, but upon release of all contacts the object animates back to its "natural" orientation (Figure 13). Since all of our omni-directional images have a clear up direction, we use the top of the sphere as the tether axis; however, any arbitrary axis can be specified as the tether axis.

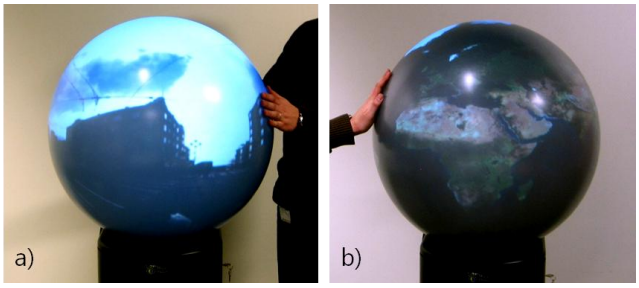

Figure 12: Examples of Sphere omni-directional visualizations: (a) panoramic walk down Seattle city street; (b) visualization of the Earth as a globe.

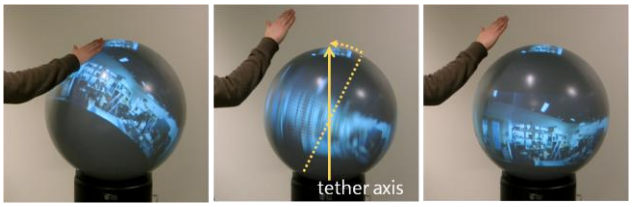

Figure 13: Tether interaction: The user can freely manipulate omni-directional video. Upon contact release, the video returns to its horizontal state.

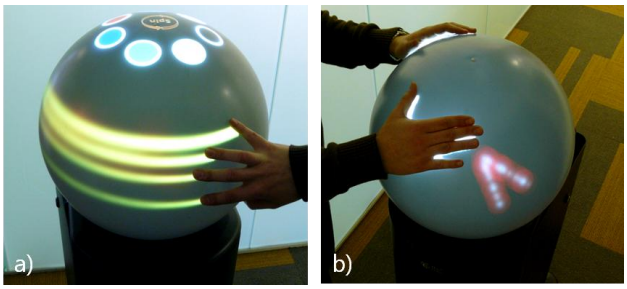

Figure 14: Two concept applications that use the entire surface contact area as input: (a) Sphere Paint, and (b) Sphere Pong.

Sphere Paint and Sphere Pong

In addition to multi-point interactions that rely on tracking individual contact points, we explored interactions that use the entire touch area as input. We designed two concept applications: Sphere Paint and Sphere Pong (Figure 14). In Sphere Paint, the user can paint on Sphere with a finger or any available object (Figure 14a). Exploiting its spherical shape, we allow Sphere to spin slightly (similar behavior to a potter's wheel), thus offering interesting artistic possibilities to the user. Sphere Pong is a game prototype in which users can use their hands or any other object to bounce several balls around the spherical surface and score points (Figure 14b). As with the classic game Battleship, not being able to see other users' actions adds an exciting dimension to an otherwise standard pong game.

INITIAL USER OBSERVATIONS

We exhibited Sphere on three occasions at high-traffic locations in our organization (Figure 15). While these events were not formal user evaluations, they presented us with an opportunity to observe hundreds of people interacting with Sphere in an informal manner with little to no instructions. We now discuss our observations of multi-user interactions during these sessions.

Sphere's unusual shape, large size, and visibility from all directions attracted large crowds. People frequently described their experience as "magical" and "like interacting with a crystal ball." Photos and videos were easily browsed by several people independently, shared with others when desired, and moved to the unused space on the display for "storage of unwanted items" (usually the bottom hemisphere). This use pattern suggests that the territoriality concepts introduced by Scott et al. [20] are applicable on Sphere, but more formal evaluation is clearly needed to explore differences caused by the form factor.

Omni-directional data spanning the entire display proved more difficult to handle when many people tried to interact with Sphere at the same time. Because a given user cannot see other users touch Sphere in the other hemisphere, users often either fought for control or became confused by movement that they did not initiate.

We also observed that people were much more inclined to start touching and interacting with Sphere if the objects were left in a disorganized, "messy" arrangement by previous users. In contrast, when the interface was reset and displayed objects were realigned, we usually had to demonstrate and explain that Sphere is indeed touch-sensitive and interactive. This observation is especially important with respect to public interactive displays which rely on an "attract mode" to solicit public touch and interaction. We suggest placing objects in a non-perfectly aligned arrangement to draw users into interaction.

During our demo sessions, we logged all touches and created radial heat maps for further analysis. These heat maps confirm that spherical displays have no master user position. While a heat map of a 15-minute single user session (Figure 16a) suggests that a user assumes a particular position and does not move extensively around Sphere, the 3-hour log of a number of different users shows no clear orientation preference (Figure 16b). In total, we examined logs of 21 hours of high-traffic use, all of which show similar user behavior. While promising, these results are clearly preliminary observations. We are planning to conduct more formal evaluations to confirm and extend these findings.

Users viewing the Earth's surface on Sphere often requested high magnification (zoom) capability. In addition to scaling problems already discussed in this paper, high zoom level effectively results in a transformation of a spherical surface data into a mostly flat map-like data, which is not well suited for display on Sphere. We plan to further investigate zooming on Sphere and to study its implications for interactivity and collaboration.

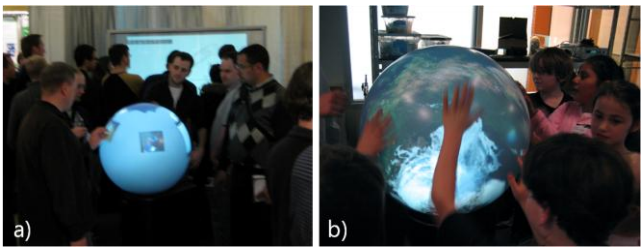

Figure 15: Collaboration on Sphere in high-traffic locations: (a) 5 adults browsing videos; (b) 7 children interacting with a globe.

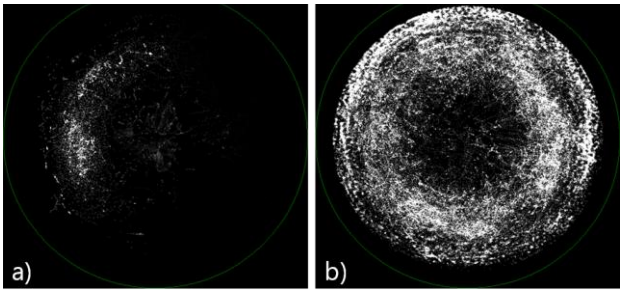

Figure 16: Radial heat maps of user touches around the 24" Sphere: (a) a single user's 15-minute session; (b) more than 50 users during a 3-hour demo session.

DISCUSSION AND CONCLUSIONS

We have presented Sphere, a novel hardware and software solution that enables multi-touch sensing on a spherical display. Our unique solution creates a self-enclosed device, capable of displaying and sensing data on a spherical surface, without occlusions and shadowing problems. To allow for rich user interaction, we implemented four concept applications and a set of multi-touch interaction techniques that account for the unique characteristics of the spherical form and assist multi-user collaboration. We also contribute several interesting observations of Sphere being used by hundreds of people in three high traffic locations.

The interactions and concepts presented in this paper form the building blocks for interactive spherical displays, but much research remains to be done. We are curious about the effect the Sphere size has on interaction as we believe that different sizes will yield different interaction paradigms (e.g., a handheld Sphere vs. a room-sized one). Collaborative scenarios interest us the most, given the clear collaborative potential of this form factor. How do people align themselves around the display to best accomplish a task? How can user-interaction conflicts resulting from omni-directional data be effectively handled? How does sphere size affect collaboration? We believe all of these questions are worthy of further research.

In addition, we are investigating using Sphere as a displayable input device for navigational control of a remote robot or of an avatar in a synthetic virtual world. A similar sensing mechanism to the one developed for Sphere could also be used to enable touch-sensing on different convex form factors (e.g., hemispheres, cylinders, cuboids).

While the work described in this paper focused on Sphere being used as a single primary device integrating both dis-

play and input sensing, it is important to not restrict our thinking to only solitary device usage scenarios. Indeed, it is highly likely that future use of Sphere will be within a broader heterogeneous ecology of displays and input technologies [6]. Rather than the "one size fits all" approach of current desktop computing, having different devices, each well suited to particular tasks, will likely as a whole provide a richer and more appropriate "workshop" for information access and manipulation. We see Sphere as an integral element in such an information workshop of the future, and resulting issues such as information transfer between display formats of vastly different form factors will continue to provide interesting research challenges.

ACKNOWLEDGMENTS

We thank Mike Foody from Global Imagination for loaning us two units of the Magic Planet display, Billy Chen and Eyal Ofek for their feedback and the omni-directional data, Mike Sinclair for his help with hardware, and Eric Horvitz and Patrick Baudisch for their brainstorming ideas.

REFERENCES

1. Bowman, D.A., Kruijff, E., LaViola, J.J., and Poupyrev, I. (2004). *3D User Interfaces: Theory and Practice*. Addison-Wesley, Boston.
2. Companje, R., van Dijk, N., Hogenbirk, H. and Mast, D. (2007). Globe4D, Time-Traveling with an Interactive Four-Dimensional Globe. *ACM MULTIMEDIA*. p. 959–960.
3. Chan, L.-W., Chuang, Y.-F., Yu, M.-C., Chao, Y.-L., Lee, M.-S., Hung, Y.-P. and Hsu, J. (2007). Gesture-based Interaction for a Magic Crystal Ball. *ACM VRST Virtual Reality Software and Technology*. p. 157–164.
4. Chen, Y., Au, J., Kazlas, P., Ritenour, A., Gates, H. and McCreary, M. (2003). Flexible Active-Matrix Electronic Ink Display. *Nature*. 423. p. 136.
5. Dietz, P. and Leigh, D. (2001). DiamondTouch: A Multi-User Touch Technology. *ACM UIST*. p. 219–226.
6. Fitzmaurice, G., Khan, A., Buxton, W., Kurtenbach, G., and Balakrishnan, R. (2003). Sentient data access via a diverse society of devices. *ACM Queue*. p. 53-62.
7. Grossman, T., Wigdor, D. and Balakrishnan, R. (2004). Multi-Finger Gestural Interaction with 3D Volumetric Displays. *ACM UIST*. p. 61–70.
8. Grossman, T. and Balakrishnan, R. (2006). The design and evaluation of selection techniques for 3D volumetric displays. *ACM UIST*. p. 3–12.
9. Han, J. (2005). Low-cost multi-touch sensing through frustrated total internal reflection. *ACM UIST*. p. 115–118.
10. Kettner, S., Madden, C. and Ziegler, R. (2004). Direct Rotational Interaction with a Spherical Projection. *Creativity & Cognition Symposium on Interaction: Systems, Practice and Theory*.
11. Kruger, R., Carpendale, S., Scott, S. and Greenberg, S. (2003). How People Use Orientation on Tables: Comprehension, Coordination and Communication. *ACM SIGGROUP Conference on Supporting Group Work*. p. 369–378.
12. Liang, J. and Green, M. (2004). JDCAD: A Highly Interactive 3D Modeling System. *Computers and Graphics. 18(4)*. p. 499–506.
13. *Magic Planet* by Global Imagination. www.globalimagination.com.
14. Marchese, F. and Rose, J. (2006). Projected Hemispherical Display with a Gestural Interface. *ACM SIGGRAPH Research Posters.*
15. Matsushita, N. and Rekimoto, J. (1997). HoloWall: Designing a Finger, Hand, Body, and Object Sensitive Wall. *ACM UIST*. p. 209–210.
16. Morris, M., Ryall, K., Shen, C., Forlines, C. and Vernier, F. (2004). Beyond "Social Protocols": Multi-User Coordination Policies for Co-located Groupware. *ACM CSCW*. p. 262–265.
17. *OmniGlobe* by ARC Science Simulations. www.arcscience.com.
18. *PufferSphere* by Pufferfish. www.pufferfishdisplays.co.uk.
19. Rekimoto, J. (2002). SmartSkin: An Infrastructure for Free-hand Manipulation on Interactive Surfaces. *ACM CHI*. p. 113–120.
20. Scott, S., Sheelagh, M., Carpandale, T. and Inkpen, K. (2004). Territoriality in Collaborative Tabletop Workspaces. *ACM CSCW*. p. 294–303.
21. Shen, C., Vernier, F., Forlines, C. and Ringel, M. (2004). DiamondSpin: An Extensible Toolkit for Around-the-Table Interaction. *ACM CHI*. p. 167–174.
22. Shoemake, K. (1985). Animating Rotation with Quaternion Curves. *ACM SIGGRAPH*. p. 245–253.
23. Shoemake, K. (1992). ARCBALL: A User Interface for Specifying Three-Dimensional Orientation Using a Mouse. *Graphics Interface*. p. 151–156.
24. Ushida, K., Harashima, H., and Ishikawa, J. (2003). i-ball2: An Interaction Platform with a Crystal-ball-like Display for Multiple Users. *International Conference on Artificial Reality and Teleexistence.*
25. Wilson, A. (2004). TouchLight: An Imaging Touch Screen and Display for Gesture-Based Interaction. *ICMI Conference on Multimodal Interfaces*. p. 69–76.
26. Wilson, A. (2005). PlayAnywhere: A Compact Tabletop Computer Vision System. *ACM UIST*. p. 83–92.
27. Wu, M. and Balakrishnan, R. (2003). Multi-Finger and Whole Hand Gestural Interaction Techniques for Multi-User Tabletop Displays. *ACM UIST*. p. 193–202.