CrystalDome: A Projected Hemispherical Display with a Gestural Interface

Francis T. Marchese, Jonas Borjesson, Josh Rose
Department of Computer Science, Pace University, New York, NY 10038
fmarchese@pace.edu

Abstract

CrystalDome is an inexpensive alternative to a volumetric display based on video projector technology that supports omni-directional viewing (Figure 1). It can show mapped spherical surfaces, such as globes, and scenes composed of three-dimensional objects. In the latter case, an algorithm was developed to perform hidden surface elimination in the hemisphere's reference frame. Participants can sit around the hemisphere, which is built into a low table. Sensors capture hand movements near the hemisphere for adjustment of the position, orientation and size of the objects that are projected from below onto the inside of the display.

1. Introduction

Volumetric dome displays [1], such as the Felix 3D [2] or Actuality Systems' Perspecta Spatial3D Display [3], provide 360 degree see-through views of three-dimensional data. The advantages of such systems are two-fold. First, they provide a viewer with the ability to interrogate structural data from all angles by allowing circumnavigation of the display. Second, they support collaboration, because viewers can assemble around the display, making it a focal-point for discussion. The disadvantages of these systems are that they are expensive, are not designed to display surface maps such as globes, and they render shaded surfaces poorly.

We have recently built an inexpensive alternative to a volumetric dome display based on video projector technology called CrystalDome that supports omni-directional viewing (Figure 1). It can show mapped spherical surfaces, such as globes, and scenes composed of three-dimensional objects. The hemisphere is built into a low table to support seated viewing from all sides. In addition, interaction with the display is through hand motion. Sensors capture hand movements near the hemisphere to permit relaxed adjustment of the position, orientation and size of the objects without touching the display.

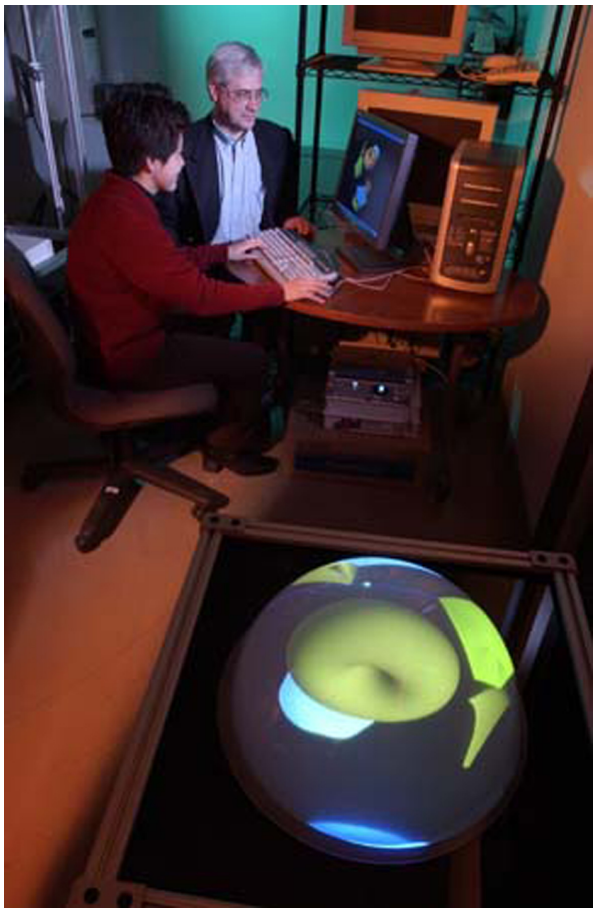

Figure 1. CrystalDome display.

There have been a number of recent approaches to building spherical and hemispherical display systems. ARC Science Simulations produces two commercially available OmniGlobe™ [4] spherical display systems of either 1.5 or 2.0 meter diameters that use internal projectors and a convex mirror to cast images that fill the sphere. ViBall [5] uses two external projectors to

11th International Conference Information Visualization (IV'07)
0-7695-2900-3/07 $20.00 © 2007 IEEE

create overlapping images that cover three-quarters of a rotatable sphere, while Globe4D [6] employs a single projector to direct a hemispherical map from above onto a freely moving sphere. These systems are designed for the display of two-dimensional content such as maps. In contrast, the DOME (Digital Object Media Environment) [7] hemispherical display is designed for interactive rendering of three-dimensional models. It uses six camera-calibrated projectors to fill a 32 inch diameter hemisphere from below. These projectors are attached to a PC rendering cluster interconnected through a gigabit Ethernet network. Each rendering client computes an image segment that is blended into a complete image and displayed.

A variety of interaction schemes have been developed for dome and globe systems. Balakrishnan et al. [8] have surveyed these issues for volumetric displays. Specific applications include Grossman et al. [9] who use motion tracking of the user's fingers on and around the display's hemispheric enclosure to support direct gestural interaction with the virtual objects. Yasuhara et al. [10] have designed a mobile PC application that controls a volumetric display from a distance, while the DOME is controlled through head-tracking devices attached to two collaborators. For globe displays, ViBall can be controlled by either mouse or hands-on movement of the sphere; and Globe4D is designed for direct hands-on control.

CrystalDome differs from the above systems in two ways. First, compared to the technologically heavy DOME multi-projector rendering cluster, it is technologically light, consisting of a single projector combined with a PC running rendering software. As such, it must be able to support omni-directional viewing without special hardware, such as that utilized in the DOME. Second, it takes a different approach to interaction with the use of sensors instead of mice or direct hands-on manipulation. Since CrystalDome is designed to be the center of seated conversation, the passing of a mouse among users was viewed as an encumbrance, so the capture of an individual's hand motion became a design requirement.

In the following section we present the technology behind CrystalDome. This is followed by results and discussion and finally suggestions for future work.

2. Technology

The hemispherical display consists of three functional components: a microcontroller-sensor system, a graphics-projection system, and interface software that runs on the PC and translates sensor data into application-specific controls.

The microcontroller-sensor system was built with three distance sensors attached to a Basic Stamp 2 microcontroller by Parallax Inc. [11]. Microcontrollers are self-contained computers that possess multiple I/O channels to control lights, motors, and sensors. They can be programmed in languages such as Basic, Java, and C. Once programmed, microcontrollers can run unaided or communicate with a PC through a standard serial or USB port. Three PING sonar sensors manufactured by Parallax (range - 2cm to 3m (~.75" to 10')) were used to control input for x-axis, y-axis and zoom. The microcontroller was programmed in Basic to send sensor id and range data to the PC's serial port.

Sensor data needed to be converted into mouse movements or keyboard controls that could be understood by application programs. There are two possible approaches. The first is to modify the application so it accepts sensor input; the second is to create interface software to translate sensor data into a form understandable by an application. We decided upon the latter approach because many software systems are only available as executables and those that do provide source code may have designs that require a significant amount of effort to learn and modify.

An interface controller was designed to read sensor data from the serial port, interpret it, issue commands that are recognized by the application programs, and keep track of the current system state. The system was implemented in Processing.org [12 - 13], an open source programming language and environment based on Java, that is used for prototyping visually oriented software and as a professional production tool. Processing provides access to many class libraries such as video and networking while hiding much of the programming details. Our interface software took advantage of a Processing class library that supports serial port communication. In addition, Java's Robot classes were included so that the interface system could generate input events for mouse and keyboard control as needed. These classes are part of Java's abstract window toolkit and provide methods such as `keyPress()`, `mouseMove()`, and `mousePress()`.

In operation, the interface controller works in two possible ways. For mouse controlled software, it translates the user's hand motion toward and any from each of the three sensors into left-right *x* cursor movement, up-down *y* cursor movement, and in-out zoom, respectively. For keyboard controlled software,

the controller issues appropriate key-pressed commands to generate the same mouse actions.

The third component of the hemispherical display is the graphics-projection system. The projection system employs a commodity video projector with XGA resolution (1024 x 768) attached to a standard video card that takes an image and projects it with the aid of an angled planar mirror onto the underside of a Plexiglas hemisphere that has been sprayed with a light-diffusing coating (c.f. Figure 1). Any two-dimensional spherical map projection (e.g. Google Earth) that appears distended at the poles and compressed at the equator will correctly map to the hemisphere with no intermediate processing, but a single point projection of a three-dimensional scene will not.

The origin of this problem is in the view dependency of each point on the surface of the hemisphere. Specifically, every point on the dome's surface is a unique viewpoint in which the observer looks into the hemisphere towards its center. Therefore, the process of projecting a three-dimensional object onto the surface of the hemisphere must employ a 3D to 2D mapping that considers the curvature of the sphere. There are a number of possible approaches to the solution of this problem.

One approach is to use cube or environment mapping [14]. In cube mapping, six cameras are placed at some station point and six pictures are taken, one each along a corresponding positive or negative Cartesian axis direction. These are then assembled into a cube and projected onto the surface of a sphere. This method works well for immersive dome environments such as planetariums or even multi-screen video games where a viewer looks away from the center of projection towards the outside world. However, with the hemisphere, the viewer looks in the opposite direction, toward the center of projection.

Ray tracing methods could be employed as an alternative approach. Here, each pixel on the hemisphere's surface is the starting point for a ray that terminates at the hemisphere's origin. The advantage of this approach is that it performs hidden surface elimination and global illumination computations in the hemisphere's reference frame. The disadvantage is that these computations are time consuming, requiring a significant amount of additional graphical resources and processing for real-time rendering.

Another possibility would be to configure a single pixel camera utilizing a hardware-supported graphics API such as OpenGL. The camera would be placed on the surface of the hemisphere pointing towards its origin. A picture would be taken at each pixel location and assembled into an image. The advantage of this method is that all hidden surface and illumination computations are performed by the graphics hardware. The disadvantage here is that for a commodity projector's resolution, it would take approximately 600K camera movements and individual snapshots per frame.

Our approach was to distort the geometry of the scene so that the points defining objects contained within it are warped along radial paths from the sphere's center to viewpoints on its surface. In so doing, the shapes of objects conform to the curvature of the space defined by a spherical projection. As a simple example consider the mapping of a rectangle to a sphere's surface as shown in Figure 2. Radial paths are traced from the sphere's origin, passing through the red rectangle, distorting its surface points into a warped grid (Figure 2a) shown on the sphere's surface (Figure 2b).

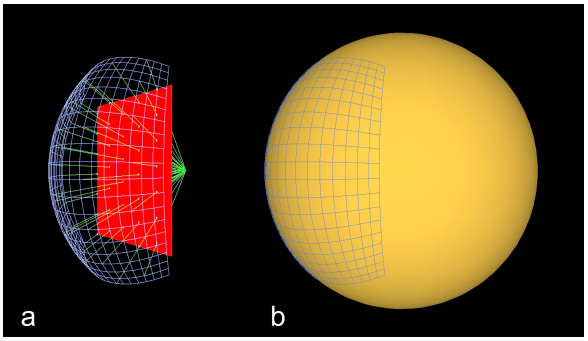

Figure 2. Projection of red rectangle to a sphere: a) radial projections, b) polygon net on sphere surface.

Our rendering algorithm does not project all points to the sphere's surface – this would flatten the scene entirely; but rather, moves each point a fraction (λ) of the distance to the sphere's surface along the radial vector defined by the sphere's center **O** and the point's starting position **P**. For $\lambda = 0$, the point remains unchanged; conversely, for $\lambda = 1$, the object is completely flattened. In order to apply λ uniformly to every object in the scene, each object is assigned its own concentric sphere, the radius of which is defined by its maximum radial extent from the origin. In addition, and for more generality, an ellipsoid is used as the bounding space instead of a sphere. The ellipsoid provides additional control over shape of the space, allowing adjustment of three radial dimensions as opposed to the sphere's one.

After object warping has been carried out, a single-point perspective projection is performed from the sphere's apex. Hidden surface elimination follows. Because the C++ application employs the OpenGL API, rendering algorithms are executed in hardware.

3. Results and Discussion

A number of mapping applications were run on CrystalDome including Google Earth [15] as well as the 3D rendering program we designed to demonstrate the warping algorithm. A portion of the Earth's northern hemisphere rendered by Goggle Earth is displayed in Figure 3. A 3D scene composed of quadrics is shown in Figure 1.

A Goggle Earth map is projected as-is, without any intermediate processing. As can be seen from Figure 3, the map regions near the equator that would normally be compressed in the two-dimensional map projection are stretched over the surface as it would appear on a globe.

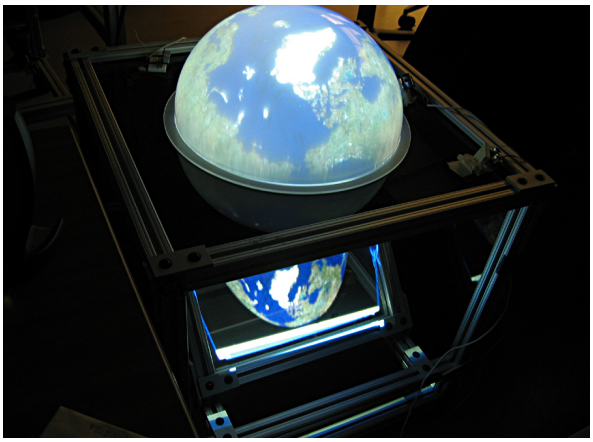

Figure 3. CrystalDome display of Google Earth.

Figure 4 shows close-up pictures of a 3D scene projected onto the CrystalDome with and without spatial warping. The top portion of the figure displays how the image appears on the computer monitor, while the lower images record what is projected onto the dome. The unwarped images are displayed on the left and warped images on the right of the figure. First, notice the warped image on the computer monitor. This distorted space is comparable to that produced by the spherical nonlinear perspective projection of Yang and coworkers' [16]. Notice also how the torus is distended in the unwarped dome projection, while the warped version ($\lambda = 0.5$) correctly projects the torus's circular form. However, the pentagonal object in the warped dome projection should have straight edges; instead these edges appear as arcs. This problem results from using the predefined quadrics built into OpenGL. Our algorithm only warps the object's control points, leaving OpenGL to do the rest. As a result, the object's straight edges are projected as arcs on the dome surface. The clear solution is to not construct scenes from geometric objects defined in the graphics API; instead, have the application software define graphical objects so it may control their degree of tessellation and hence the the number of surface points to be warped [17].

The sensor system was set up so all three sensors were aligned along the edge of the dome framework (c.f. Figure 5). Left and right sensors controlled rotation, while the center sensor controlled zoom. It was found that in order for the sensors to provide smooth motion control with a variety mouse driven software, it was necessary to calibrate the interface software for each system to accommodate different mouse speeds and sensitivities. This was the case as

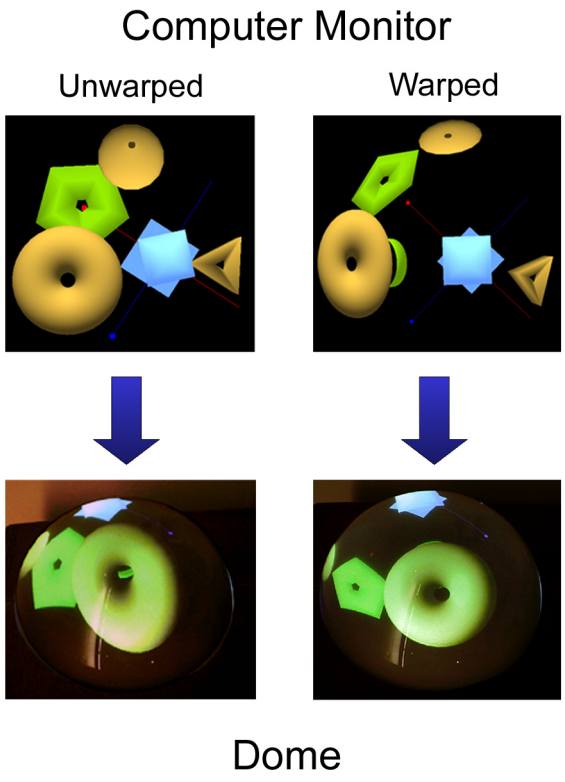

Figure 4. Dome display without image warping (left) and with image warping (right).

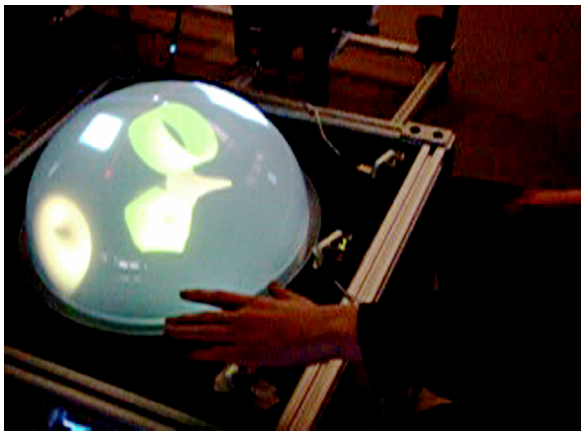

Figure 5. Sensor system on top of dome frame with operator's hand over left sensor.

well with the 3D rendering program that was controlled by keyboard commands.

Overall, the hemispherical display worked remarkably well for rendering three-dimensional scenes. The projection algorithm creates a clear sense of depth within the display. Indeed, an animated scene gives the feeling that objects are floating within a glass sphere. The system is less successful at producing a sense of transparency and depth, characteristics intrinsic to true volumetric displays, but objects rendered in the hemisphere maintain a better sense of solidity, light and shadow.

4. Future Work

There are a number of areas in which this research may be extended. For example, CrystalDome's original design called for sensors to be placed in such a way as to allow multiple users seated around the display to control the movement of its contents. As such, the number of sensors, their placement, and the types of gestural interactions employed to control CrystalDome remain to be optimized. In addition, the three-dimensional scene that was used to demonstrate the spatial warping algorithm was limited to a few kinds of geometric objects. In order for CrystalDome to be useful in a wide range of visualization applications, it must support additional geometric primitives such as polygon meshes and nurbs. In so doing, the algorithm must render spatially warped objects with a sufficient degree of tessellation to map properly to the hemisphere, but not overly tessellated to waste resources. These are areas of continued research.

References

[1] G.E. Favalora, "Volumetric 3D displays and application infrastructure," *IEEE Computer* 38, 8, 2005, pp. 37-44.

[2] K. Langhans, D. Bezecny, D. Homann, D. Bahr, K. Oltmann, K. Oltmann, C. Guill, E. Rieper, and G. Ardey, "FELIX 3D Display: An interactive tool for volumetric imaging," *Stereoscopic Displays and Virtual Reality Systems IX*, *Proceedings of SPIE*, vol. 4660, San Jose, CA, 2002.

[3] W-S. Chun, J. Napoli, O.S. Cossairt, R.K. Dorval, D.M. Hall, T.J. Purtell II, J.F. Schooler, Y. Banker, and G.E. Favalora, "Spatial 3-D infrastructure: display-independent software framework, high-Speed rendering electronics, and several new displays," *Stereoscopic Displays and Virtual Reality Systems XII*, A.J. Woods, M.T. Bolas, J.O. Merritt, I.E. McDowall (eds.), *Proceedings of SPIE-IS&T Electronic Imaging*, *SPIE* vol. 5664, 2005, pp. 302-312.

[4] OmniGlobe™, ARC Science Simulations, [cited Feb. 21, 2007], available from World Wide Web: <http://www.arcscience.com/omni.htm >.

[5] S. Kettner, C. Madden, and R. Ziegler, "Direct rotational interaction with a spherical projection", *Interaction: System, Practice and Theory*, ACM SIGCHI, 2004.

[6] R. Companje, N. van Dijk, H. Hogenbirk, and D. Mast, "Globe4D: time-traveling with an interactive four-dimensional globe," In *Proceedings of the 14th Annual ACM international Conference on Multimedia* (Santa Barbara, CA, USA, October 23 - 27, 2006). ACM Press, New York, NY, pp. 959-960.

[7] S. Webb and C. Jaynes, "The DOME: A portable multi-projector visualization system for digital artifacts", *IEEE Workshop on Emerging Display Technologies*, Bonn, Germany, 2005.

[8] R. Balakrishnan, G.W. Fitzmaurice, and G. Kurtenbach, "User interfaces for volumetric displays," Computer 34, 3 (Mar. 2001), pp. 37-45.

[9] T. Grossman, D. Wigdor, and R. Balakrishnan, "Multi-finger gestural interaction with 3D volumetric displays." In *Proceedings of UIST '04*, (October 24–27, 2004) , Santa Fe, New Mexico, pp. 61-70.

[10] Y. Yasuhara, N. Sakamoto, N. Kukimoto, Y. Ebara, and K. Koyamada, "Interactive controller for 3D contents with omni-directional display," In *Proceedings of ICPADS*, 2005, pp. 167-171.

[11] Basic Stamp Microcontroller, Parallax, Inc., [cited Feb. 21, 2007], available from World Wide Web: <http://www.parallax.com/html_pages/products/basicstamps/basic_stamps.asp >.

[12] C. Reas and B. Fry, "Processing: programming for the media arts. *AI Soc.* 20, 4 (Aug. 2006), pp. 526-538.

[13] Processing, [cited Feb. 21, 2007], available from World Wide Web: < http://processing.org/ >.

[14] N. Greene, "Environment mapping and other applications of world projections." *IEEE Comput. Graph. Appl.* 6, 11 (Nov. 1986), pp. 21-29.

[15] Google Earth, [cited Feb. 21, 2007], available from World Wide Web: < http://earth.google.com/ >.

[16] Y. Yang, J.X. Chen, and M. Beheshti, "Nonlinear perspective projections and magic lenses: 3D view deformation," *IEEE Comput. Graph. Appl.* 25, 1 (Jan. 2005), pp. 76-84.

[17] T.K. Heok and D. Daman, "A review on level of detail," In *Proceedings of the International Conference on Computer Graphics, Imaging and Visualization (Cgiv'04),* (July 26 - 29, 2004). IEEE Computer Society, Washington, DC, pp. 70-75.