The Virtual Showcase

Oliver Bimber
Fraunhofer Institute for Computer Graphics, Germany

Bernd Fröhlich
Bauhaus University Weimar, Germany

Dieter Schmalstieg
Vienna University of Technology, Austria

L. Miguel Encarnação
Fraunhofer Center for Research in Computer Graphics, USA

We present the Virtual Showcase, a new multiviewer augmented reality display device that has the same form factor as a real showcase traditionally used for museum exhibits.

Intuitive access to information in habitual real-world environments is a challenge for information technology. An important question is how can we enhance established and well-functioning everyday environments rather than replace them by virtual environments (VEs)? Augmented reality (AR) technology has a lot of potential in this respect because it augments real-world environments with computer-generated imagery. Today, most AR systems use see-through head-mounted displays, which share most of the disadvantages of other head-attached display devices.

In this article, we introduce a new projection-based AR display system—the Virtual Showcase (see Figure 1). The Virtual Showcase has the same form factor as a real showcase, making it compatible with traditional museum displays. Real scientific and cultural artifacts are placed inside the Virtual Showcase allowing their 3D graphical augmentation. Inside the Virtual Showcase, virtual representations and real artifacts share the same space providing new ways of merging and exploring real and virtual content. Solely virtual exhibits may also be displayed. The virtual part of the showcase can react in various ways to a visitor, which provides the possibility for intuitive interaction with the displayed content.

Another interesting aspect of our system is its support for two to four simultaneously tracked users looking at the Virtual Showcase from different sides. This feature allows the collaborative exploration of artifacts shown in the Virtual Showcase. These interactive showcases contribute to ambient intelligent landscapes, where the computer acts as an intelligent server in the background and visitors can focus on exploring the exhibited content rather than on operating computers.

As Figure 1 shows, the Virtual Showcase consists of two main parts (the numbers in parentheses correspond to the numbers in the figure): a convex assembly of half-silvered mirrors (1) and a graphics display (2). So far, we've built Virtual Showcases with two different mirror configurations. Our first prototype consists of four half-silvered mirrors assembled as a truncated pyramid (see Figure 1a). Our second prototype uses a single mirror sheet to form a truncated cone (see Figure 1b). We placed these mirror assemblies on top of a projection screen (2). Users can see real objects inside the showcase through the half-silvered mirrors merged with the graphics displayed on the projection screen. We illuminated the showcases' contents with a controllable light source (3) while presenting view-dependent stereoscopic graphics to the observer. For our current prototypes, we use standard shutter glasses (5) controlled by infrared emitters (4). Head tracking is accomplished using an electromagnetic tracking device (6).

Our pyramid-shaped prototype supports up to four viewers simultaneously looking at the showcase from four different sides. Our cone-shaped prototype provides a seamless surround view onto the displayed artifact.

We renderend the scenes in this article on a 500-MHz Pentium III with nVidia Quadro2 graphics. Note that we haven't touched up the photographs. They appear as seen from the viewer's perspective and we rendered them as monoscopic images, although the rendering algorithms normally produce stereoscopic images.

Background and related work

Here we discuss two main areas related to our approach: display technology for AR systems and mirror display technology. In addition, we outline our approach from a geometric optics point of view.

Display technology for augmented reality

Optical or video see-through head-mounted displays (HMDs) are currently the display devices mainly used for AR. However, we can attribute several disadvantages to HMDs:

- deficient resolution and field of view (FOV),
- ergonomic drawbacks due to heavy and cumbersome devices, and

0272-1716/01/$10.00 © 2001 IEEE

- visual perception issues that result from constrained focal lengths.

These disadvantages call for the development of alternative display concepts.

Spatially AR[1] represents such an alternative. Front-projection devices seamlessly project images directly on physical objects' surfaces instead of displaying them somewhere else in the viewer's visual field. On the one hand, this overcomes some of the drawbacks related to HMDs. On the other hand, Spatially AR introduces new problems such as

- shadow-casting of the physical objects and interacting users that results from using front projection and
- restrictions on the display area, which is constrained to the physical objects' size, shape, and color.

Mirror displays

Besides optical see-through HMDs, a few other display systems exist that apply full or half-silvered mirrors to reflect screens. We can categorize them into the following classes: Pepper's Ghosts configurations,[2] reach-in systems,[3-5] real-image displays,[6,7] and varifocal mirror systems.[7,8] All these systems differ from our Virtual Showcase approach.

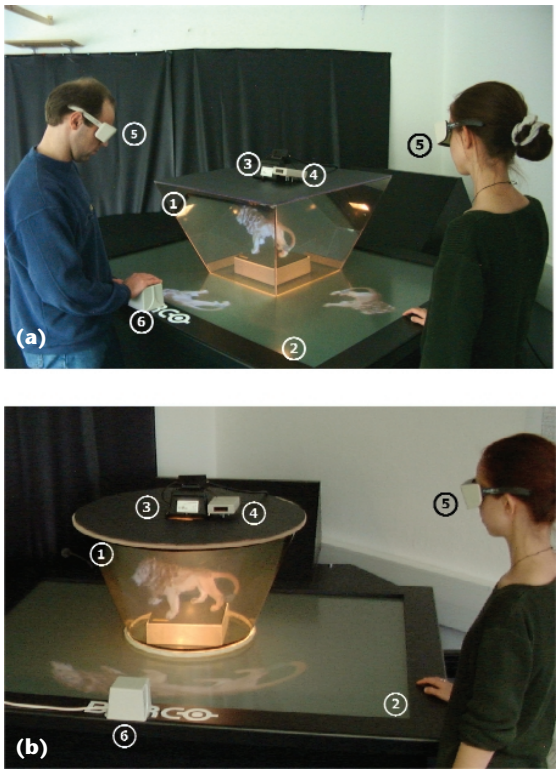
For such systems, the graphics must be transformed before being displayed to ensure that the viewers don't see mirrored or distorted images. For Pepper's Ghosts configurations and reach-in systems, this transformation is trivial (for example, a simple mirror transformation of the frame-buffer content[3] or of the world-coordinate axes[4,5]) because they constrain viewing to restricted areas and benefit from a static mechanical mirror-screen alignment. Some approaches[4] combine this transformation with the device-to-world transformation of the input device by computing a composition map during a calibration procedure and multiplying it with the device-coordinates during the application. Other approaches[5] determine the projection of virtual points on the reflected image plane via ray tracing and then map it to the corresponding frame-buffer location by reversing one coordinate component. Mirror displays that apply curved mirrors, such as real-image displays and varifocal mirror systems, generally don't predistort the graphics before they're displayed. Yet some systems apply additional optics, such as lenses, to stretch the reflected image.[7]

However, if a view-dependent rendering is required or if the mirror optics are more complex and don't require a strict mechanical alignment, these transformations become more complicated. Note that both issues apply to Virtual Showcases.

The challenge for the Virtual Showcase is developing appropriate real-time rendering methods and image deformation techniques that support a view-dependent image presentation for single or multiple users and for planar and curved mirror optics.

Our approach

We refer to the real area in front of a mirror as the *object space* and call the virtual area behind a mirror (that's perceived while looking at it) the *image space*.



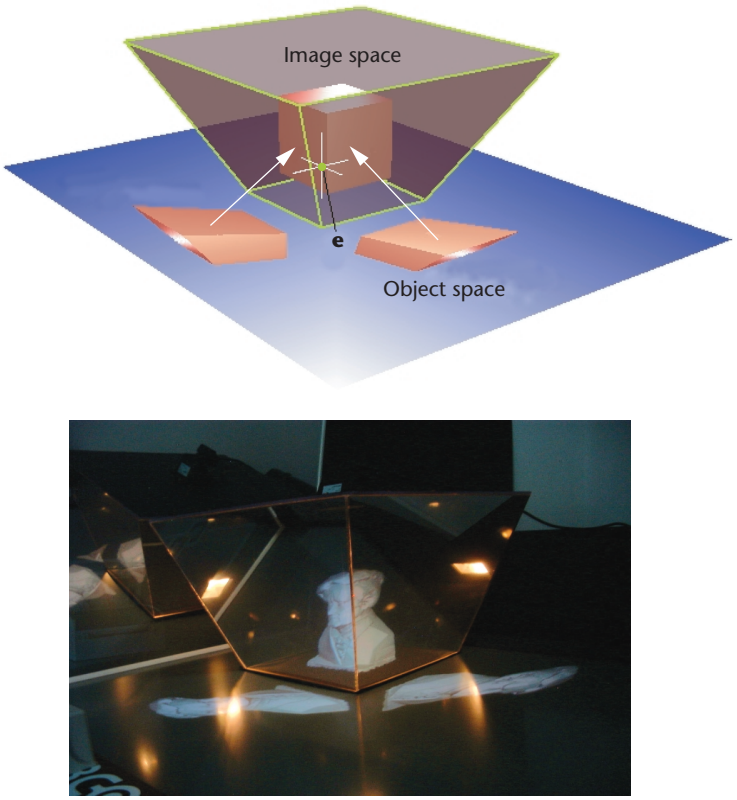
1 The Virtual Showcase prototypes. (a) The pyramid-shaped prototype that supports up to four viewers, and (b) the cone-shaped prototype that supports multiple users and provides a seamless surround-view.

Note that these definitions meet more of the conventions of geometric optics than of computer graphics, where the object space is usually the 3D world-coordinate system and the image space is the 2D projection. Depending on the mirror geometry (for example, planar or curved), the reflection of the object space maps differently into the image space. While the image space of planar mirrors is a stigmatic and affine map of the object space, the image space of curved mirrors is curvilinearly transformed. However, users don't expect to see a mirror while looking at the Virtual Showcase device. Rather, it must appear transparent—like a traditional showcase. Reflections on its surface must be seamlessly combined with the enclosed real objects.

In the case of half-silvered mirrors, the image space unites the reflected image of the object space in front of the mirror and the transmitted image of the real environment behind the mirror.

The starting point for the following rendering techniques requires a description of the scene geometry that's virtually located inside the showcase. Since, from an optics point of view, this geometry is spatially located within the image space, we call it the *image space geometry*.

We aim to transform the image space geometry into the object space in such a way that the reflection of the displayed object space optically results in the expected image space. Therefore, the transformed image space geometry is displayed on a projection plane that's located within the object space (that is, in front of the mirror). Thus, the mirror's reflection neutralizes the image space geometry's transformation. The optically formed image appears orthoscopic, and stereoscopically and perspectively correct and undistorted to an observer.



2 Multiple reflections merged into a single, consistent image space.

Because virtual and real objects coexist in conjunction within the image space, the appearance of the entire image space is known for every given viewpoint.

In the following sections, we describe rendering techniques that can be applied for Virtual Showcases built from planar mirror sections and for showcases built from a single curved mirror piece. We assume that a single planar display device (such as a rear-projection system) is used and that the display device and the mirror optics are defined within the same world-coordinate system. In the upcoming examples, the projection plane matches the world-coordinate system's x/y plane. Note that we illustrate the following methods in an OpenGL context.

Virtual showcases built from planar sections

To transform the known image space geometry appropriately into the object space, we can apply different, slightly modified transformation pipelines for each planar mirror section. With known plane parameters ($\mathbf{n}_r = [a_r, b_r, c_r]$, δ_r) for each mirror, we have to integrate two modifications into the common model-view transformation:[9]

- We apply an additional model transformation between scene transformation $\boldsymbol{M}$ (that is, the accumulation of glTranslate, glRotate, glScale, and so on) and viewpoint transformation $\boldsymbol{V}$ (such as gluLookAt). We do this by multiplying the reflection matrix $\boldsymbol{R}$ with the current transformation matrix—before viewpoint transformation and after scene transformation.
- We apply the common viewpoint transformation matrix $\boldsymbol{V}$ with the reflected viewpoint $\mathbf{e}'$ instead of the actual viewpoint $\mathbf{e}$. We compute the reflected viewpoint by transforming the actual viewpoint over the specific mirror-plane: $\mathbf{e}' = \boldsymbol{R} \cdot \mathbf{e}$.

The reflection matrix is given by

$$\boldsymbol{R} = \begin{bmatrix} 1-2a_r^2 & -2a_r b_r & -2a_r c_r & -2a_r \delta_r \\ -2a_r b_r & 1-2b_r^2 & -2b_r c_r & -2b_r \delta_r \\ -2a_r c_r & -2b_r c_r & 1-2c_r^2 & -2c_r \delta_r \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Note that the inverse of $\boldsymbol{R}$ is equivalent to $\boldsymbol{R}$ if $\mathbf{n}_r = [a_r, b_r, c_r]$ is normalized. We can then write the accumulated transformation matrix as $\boldsymbol{P} \cdot \boldsymbol{V} \cdot \boldsymbol{R} \cdot \boldsymbol{M}$, where $\boldsymbol{P}$ denotes the transformation matrix of the applied off-axis perspective projection (for example, generated by glFrustum).

Since an individual reflection matrix exists for each mirror plane, we must apply a modified model-view transformation (with individual $\boldsymbol{R}$ and $\mathbf{e}'$) for each front-facing mirror, respectively. Thus, for a given viewpoint $\mathbf{e}$, the image-space geometry is transformed and rendered multiple times (for each front-facing mirror individually). The example in Figure 2 illustrates this for a truncated pyramid-like Virtual Showcase, assuming that a single observer sees through two mirrors simultaneously.

Because the application of $\boldsymbol{R}$ also reverses the polygon order (which influences front-face determination, lighting, back-face culling, and so on), we have to reverse the polygon order explicitly between transformation and rendering.[9]

A single object displayed in the object space appears exactly once in the image space because convex mirror assemblies unequivocally subdivide the object space into individual reflection zones for each mirror, which don't intersect or overlap. Consequently, convex mirror assemblies provide a definite one-to-one mapping between the object and image spaces.

Observed from $\mathbf{e}$, the different images optically merge into a single consistent image space by reflecting the projection plane, whereby this image space visually equals the image of the untransformed image-space geometry.

The views in Figure 3 can be seen by a single viewer while moving around the showcase, or by two individual viewers while looking at different mirrors simultaneously.

Figure 4 shows an example of an augmented real exhibit, displayed within a Virtual Showcase. We textured and partially projected a virtual representation of the Buddha onto the real statue to demonstrate the precise superimposition of the two environments. We visualized additional multimedia information such as images, movies, and text annotations to complement the exhibit.

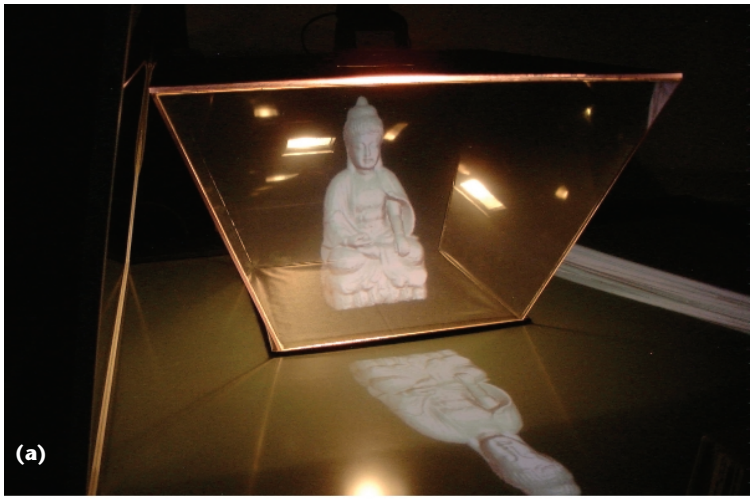
In terms of generating stereoscopic images, we must apply all transformation and rendering steps individually for each eye position of each viewer. This means that to serve four viewers simultaneously, for instance, we must

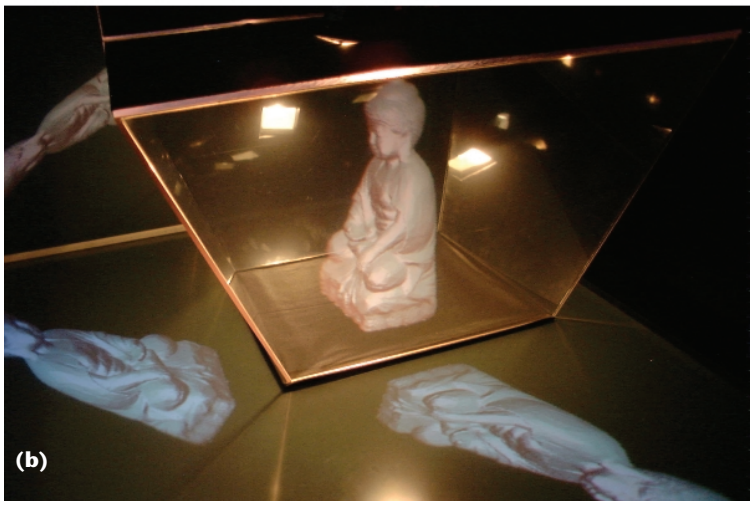
split the transformation pipeline into four subpipes after a common scene transformation $\boldsymbol{M}$. Following the application of the mirror-specific reflection transformations $\boldsymbol{R}$, we can split the subpipelines again to generate the different stereoscopic images for each eye. The subsequent eight subpipelines use different viewpoint transformations with individually reflected viewpoints $\mathbf{e}_i'$, corresponding to each eye position $\mathbf{e}_i$.

Diverging from the example described, we can present individual scenes (that is, different image spaces) to each viewer. In this case, we must apply an individual scene transformation $\boldsymbol{M}$ within each subpipe.

Because of the independence among the transformation subpipes, we can apply parallel rendering techniques (such as using multipipeline architectures). Since $\boldsymbol{R}$ is affine, the modified transformation pipeline doesn't require an explicit access to the image-space geometry. Thus, we can realize it completely independent of the application and even implement it in hardware.

To provide a single, consistent image space (consistent regardless of the viewpoint), we must register the mirror planes precisely. Slight registration errors optically result in gaps between the reflections and therefore in multiple inconsistent image spaces. This problem is comparable to misregistered multiplane projection devices or tiled displays, although for mirror planes the optical aberrations are view dependent.





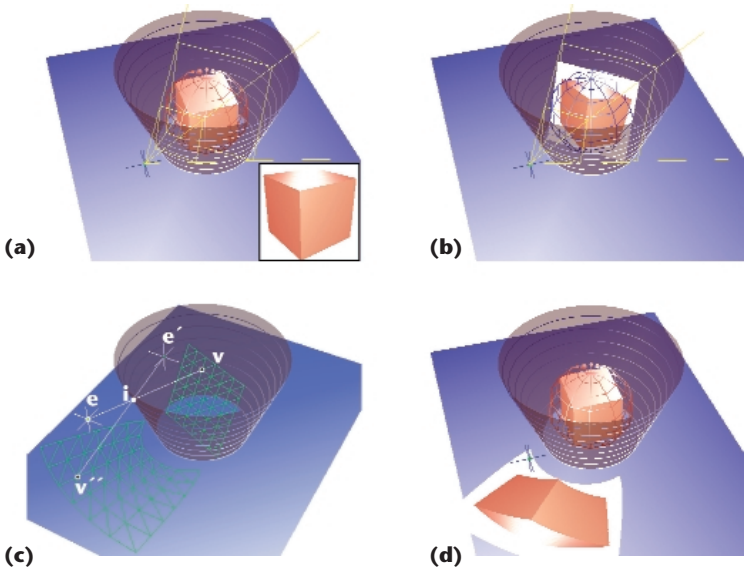
3 Two individual views into the same image space from different perspectives.

4 Augmented real exhibit displayed within a Virtual Showcase.

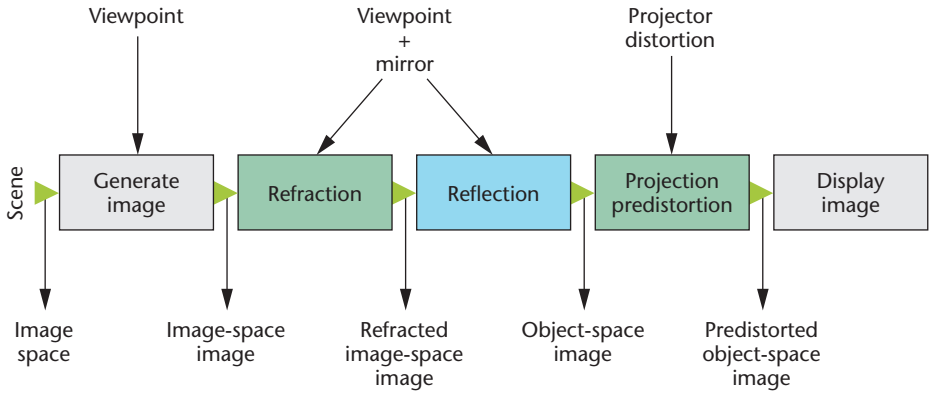
Convexly curved virtual showcases

Building Virtual Showcases from a single sheet instead of using multiple planar sections reduces the calibration problem to a single registration step and consequently decreases the error sources. In addition, the edges of adjacent mirror sections, which can be annoying in some applications, disappear.

However, using curved mirrors introduces a new problem: The transformation of the image space geometry into the object space isn't affine but curvilinear. To map the image-space geometry appropriately into the object space, curved mirrors require per-vertex viewpoint and model transformations. Therefore, we can apply a similar accumulation of transformation matrices as we do for planar mirrors. However, the matrices' parameters differ for each vertex. We've developed several nonaffine geometry transformation techniques for curved mirrors. However, we can transform only highly tessellated image-space geometries with such methods to provide an acceptable curvilinearity deformation behavior.



5 Image-based multipass method for curved optics. (a) Image generation during the first rendering pass, (b, c) image deformation, and (d) image rendering during the second pass.



6 Overview of rendering (blue) and image deformation (green) steps, expressed as a pipeline.

The following two-pass rendering method (see Figure 5) avoids direct access to the image-space geometry and consequently prevents the time-costly transformations of many scene vertices. Our method applies a sequence of intermediate nonaffine image deformations, which we currently consider most efficient for curved mirror displays. It represents a mixture between the extended camera concept[10] and projective textures.[11] While projective textures use a perspective texture matrix to map projection-surface vertices into texture coordinates of those pixels that project onto these vertices, our method projects image vertices directly on the projection surface while the texture coordinate of each image vertex remains constant. This is necessary because curved mirrors yield a different projection (that is, a different projection origin or viewpoint) for each pixel.

The main difference between our approach and the extended camera concept is that the extended camera concept generates a deformed image via ray tracing—that is, each pixel is generated from a modified primary ray. Our method deforms an existing image by projecting it individually for each pixel.

Figure 6 gives an overview of the rendering and image-deformation steps in the form of a rendering pipeline. We'll describe the first and second rendering pass, as well as the reflection transformation, in detail within the following sections. In addition to these components, we built further steps that correct optical distortion caused by in–out refraction and miscalibrated projection systems based on geometric models of these sources of error into this pipeline.

Image generation

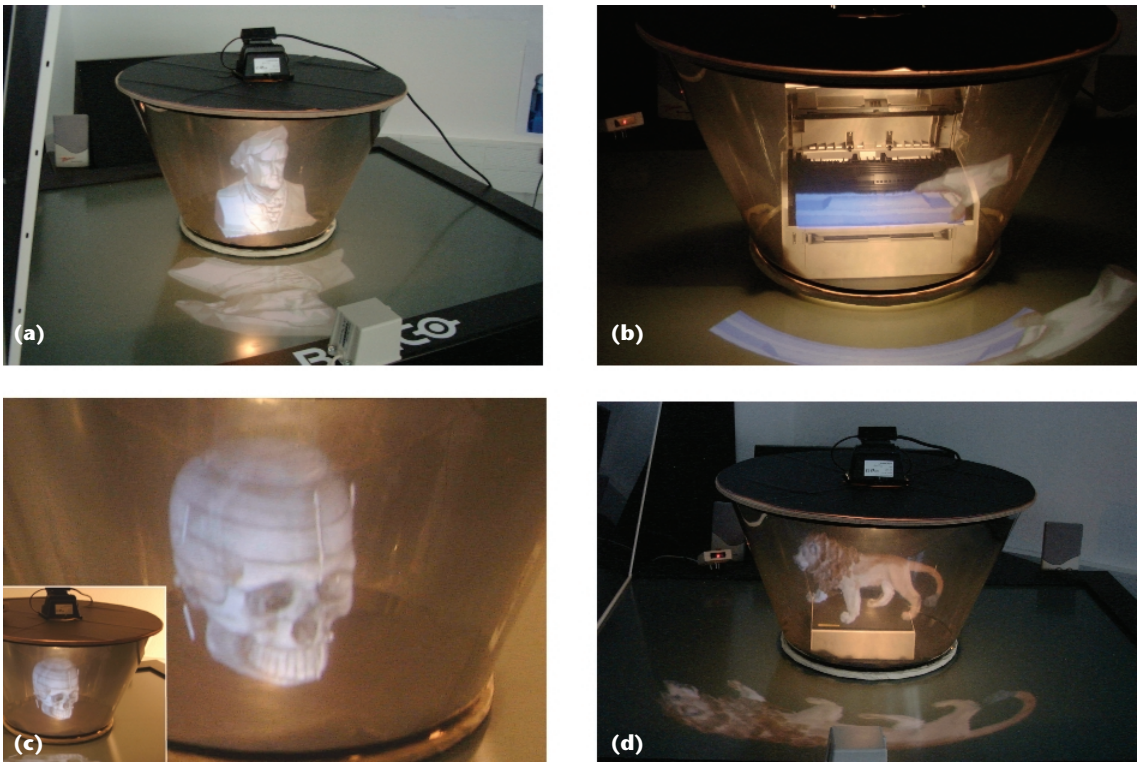
The first rendering pass creates a picture of the image space and renders it into the texture buffer rather than into the frame buffer. To generate this image, we perform an on-axis projection. We determine the size of the projection's viewing frustrum from the image space's bounding sphere. Figure 5a illustrates this for a truncated cone-like Virtual Showcase.

Note that we can exchange the first-pass rendering method. Instead of using a geometric renderer, we can use other techniques (such as image-based and nonphotorealistic rendering, interactive ray tracing, volume rendering, and so on) to generate the image space picture. Besides an ordinary geometric renderer that we used to generate the images in Figures 7a and 7b, Figure 7c shows the application of a volumetric renderer. For the image displayed in Figure 7d, however, we chose a progressive point-based renderer.

Figures 7a to 7d show some results of our rendering approach for curved Virtual Showcases, applying different image-generation methods during the first pass. While Figures 7a and 7c show exclusive virtual exhibits, Figures 7b and 7d illustrate hybrid exhibits (a virtual hand places a virtual cartridge into a real printer and a virtual lion on top of a real base, respectively).

Image geometry and reflection transformation

We now have to transform the image that we generated during the first rendering pass so that its reflection is perceived undistorted within the mirror. To support the subsequent image deformations, we must generate a geometric representation of the image plane. This image geometry consists of a uniformly tessellated grid (represented by an indexed triangle mesh) that's transformed into the current viewing frustum inside the image space so that, if the image is mapped onto the grid, each line of sight intersects its corresponding pixel (see Figure 5b). Therefore, the image is perpendicular to the optical axis and centered with the image space geometry. Finally, we transform each grid point with



7 Examples for different image generation methods: (a, b) geometric rendering, (c) volumetric rendering, and (d) point-based rendering.

respect to the mirror geometry, the current viewpoint, and the projection plane and texture each grid point with the image that we generated during the first rendering pass (see Figure 5c).

For the reflection transformation, we make sure that only visible triangles (that is, the ones with three visible vertices) are rendered during the second rendering pass. Therefore, we set a marker flag for each vertex.

For all grid vertices $\mathbf{v}$, we compute the intersection $\mathbf{i}$ of the geometric line of sight with the mirror geometry (that is, the ray that's spanned by the eye $\mathbf{e}$ and the vertex $\mathbf{v}$). Next, we determine the normal vector $\mathbf{n}$ at the intersection. The intersection point, together with the normal vector, gives the tangential plane at the intersection. Thus, they deliver the plane parameters for the per-vertex reflection transformation that can also be represented by the reflection matrix $\boldsymbol{R}$. We then use the reflection matrix to compute the reflected viewpoint $\mathbf{e}' = \boldsymbol{R} \cdot \mathbf{e}$ and the reflected vertex $\mathbf{v}' = \boldsymbol{R} \cdot \mathbf{v}$. Note that an intersection isn't given if $\mathbf{e}$ and $\mathbf{v}$ are located on the same side of a tangential plane.

Next, we generate a projection matrix $\boldsymbol{P}$ that, given a projection origin and plane parameters, projects a 3D vertex onto an arbitrary plane. In contrast to the projection for planar mirrors, only the beam that projects a single reflected vertex onto the projection plane is of interest. Thus, generating and applying a perspective projection defined by an entire viewing frustum in combination with the corresponding viewpoint transformation (such as glFrustum and gluLookAt) would implicate too much computational overhead. Consequently, this would slow down the image-deformation process. In addition, the viewpoint's reflection becomes superfluous. Since $\mathbf{e}'$, the intersection $\mathbf{i}$, and the final projection of $\mathbf{v}'$ lie on the same beam, we can use $\mathbf{i}$ as the projection origin instead of $\mathbf{e}'$. In doing this, we save the matrix multiplication for the viewpoint transformation and the determination of the viewing-frustum.

The projection matrix is given by

$$\boldsymbol{P} = \begin{bmatrix} \kappa - a_p x & -b_p x & -c_p x & -\delta_p x \\ -a_p y & \kappa - b_p y & -c_p y & -\delta_p y \\ -a_p z & -b_p z & \kappa - c_p z & -\delta_p z \\ -a_p & -b_p & -c_p & -\delta_p \end{bmatrix}$$

where $(\mathbf{n}_p = [a_p, b_p, c_p], \delta_p)$ are parameters of the projection plane, $[x, y, z, 1]$ are the coordinates of the projection center, and $\kappa = [a_p, b_p, c_p, \delta_p] \cdot [x, y, z, 1]$.

Finally, $\mathbf{v}'$ is projected with $\mathbf{v}'' = \boldsymbol{P} \cdot \mathbf{v}'$. Since $\boldsymbol{P}$ is a perspective projection, we must perform a perspective division to produce the correct device coordinates. We can summarize the entire vertex-individual transformation with $\mathbf{v}'' = \boldsymbol{P} \cdot \boldsymbol{R} \cdot \boldsymbol{M} \cdot \mathbf{v}$, whereby $\boldsymbol{M}$ denotes a possible scene transformation. Doing this for all image vertices results in the projected image within the object space (see Figure 5c).

Note that standard graphics pipelines (such as the one implemented within the OpenGL package) only support primitive-based transformations and not per-vertex transformations. Thus, we explicitly reimplemented the transformation pipeline for this approach, bypassing the OpenGL pipeline. Note also that in contrast to the transformation of scene geometry, depth handling isn't required for the image geometry's transformation

Having a geometric representation to approximate the showcase's shape (such as a triangle mesh) provides a

flexible way of describing the showcase's dimensions. However, the computational cost of the per-vertex transformations increases with a higher resolution showcase geometry. For triangle meshes, a fast ray-triangle intersection method[12] is required that also delivers the barycentric coordinates of the intersection within a triangle. We can then use the barycentric coordinates to interpolate between a triangle's three vertex normals and to approximate the normal vector at the intersection.

A more efficient way of describing the showcase's dimensions is to apply an explicit function. We can use this function to calculate the intersections and normal vectors (using its first-order derivatives) with an unlimited resolution. However, we can't express all showcase shapes by explicit functions. Since cones are simple second-order surfaces, we can use an explicit function and its first-order derivative to describe the extensions of our curved showcase. After we transform a geometric line of sight from the world-coordinate into the cone-coordinate system, we can intersect it easily with the cone by solving a quadratic equation created by inserting a parametric ray representation into the cone equation. We compute the normals by inserting the intersection points into the first-order derivative.

Image rendering

During the second rendering pass, the transformed image geometry is finally displayed within the object space, mapping the outcome of the first rendering pass as texture onto its surface (see Figure 5d). Note that we only rendered triangles that provide three visible vertices.

Since the reflection transformations of the previous step delivers device coordinates, and we've defined the projection device and the mirror optics within our world-coordinate system, a second projection transformation (such as glFrustum) and the corresponding perspective divisions and viewpoint transformation (such as gluLookAt) aren't required. If we use a plane projection device, a simple scale transformation suffices to normalize the device coordinates—for example, glScale(1/device_width/2),1/device_height/2,1). A subsequent view-port transformation finally upscales them into the window-coordinate system—for example, glViewport(0,0,window_ width, window_height).

Time-consuming rendering operations that aren't required to display the 2D image (such as illumination computations, back-face culling, depth buffering, and so on) should be disabled to increase the rendering performance. In this case, the polygon order doesn't need to be reversed before rendering, as we noted previously.

Obviously, we have the choice between a numerical and an analytical approach to intersect rays with simple mirror surfaces. Higher order curved mirrors require numerical approximations. In addition, the grid resolution that's required for the image geometry also depends on the mirror's shape. Pixels between the triangles of the deformed image mesh are linearly approximated during rasterization (that is, after the second rendering pass). Thus, some image portions stretch the texture while others compress it. This results in different regional image resolutions. However, our experiments showed that because of the symmetry of our mirror setups, a regular grid resolution and a uniform image resolution achieve acceptable image quality. Since a primitive-based (or fragment-based) antialiasing doesn't apply in deformed texture cases, we can use bilinear or trilinear texture filters instead. As with antialiasing, texture filtering is usually supported by the graphics hardware.

Note that the image's background and the empty area on the projection plane must be rendered in black, because black doesn't emit light and therefore won't be reflected into the image space. For cases in which multiple images must be composed to support multiuser applications, the black background must be color-blended appropriately.

Discussion and future work

Compared to HMDs, using Virtual Showcases for AR tasks provides high and scalable resolution and improved ergonomics because of the lightweight glasses used. In addition, since the reflected graphics appear near their real-world locations, Virtual Showcases support an easier eye accommodation.

The techniques we presented are flexible enough to be smoothly integrated into existing software frameworks and they're general enough to support different hardware setups (that is, projection devices and mirror configurations). Additionally, they take advantage of hardware-implemented rendering pipelines as much as possible. While the rendering passes and per-primitive transformations can be completely executed by graphics accelerators of today's graphics adapters, intermediate per-vertex transformations aren't supported by prevalent rendering pipelines (such as the one implemented in OpenGL). Consequently, they can't benefit from current graphics acceleration hardware. Presently, we've realized these transformations in software—that is, they tax the main CPU and memory bandwidth. Next-generation graphics engines support programmable per-vertex operations, which will allow hardware acceleration of the required per-vertex transformations.

Rather than using a uniform image tessellation, adaptive sampling approaches (such as those derived from view-dependent progressive meshes with frame-to-frame coherence) might result in additional performance resources. We plan to evaluate these techniques and their adaptation to our problem.

We also plan to evaluate other first-pass rendering techniques that generate realistic images of complex scenes and animations at interactive rates. Besides the already integrated volume rendering and splatting algorithms, image-based methods are of particular interest, because they allow realistic display of virtual representations of real artifacts.

Our current prototype uses a dimmable light source for illuminating real artifacts placed inside the showcase. This illumination causes interreflections, which become visible from the outside in some cases. We can solve this problem with upside-down configurations of Virtual Showcases (for example, using a ceiling projection). Additionally, this prevents the observers from directly seeing the projection plane. Furthermore, using a video-projector-based illumination instead of a simple light source (similar to Raskar et al.[1]) allows con-

trolled illumination on a per-pixel basis.

We need to develop interaction techniques and input devices for Virtual Showcases that are unobtrusive and work well for novice users. Since direct interaction is physically restricted by the mirror surfaces, indirect techniques and passive, real-world props seem more appropriate. These interface issues are the most challenging problems for integrating the Virtual Showcase technology into museums and other everyday environments. ■

Acknowledgments

We thank Steve Feiner and David Zeltzer for fruitful discussions and for pointing out additional related work. Thanks to Marc Levoy and Peter Neugebauer for providing and scanning the 3D models. This project is supported by the European Union IST-2001-28610, action line III.1.6 (virtual representation of cultural and scientific objects). Patent pending.

References

1. R. Raskar, G. Welch, and H. Fuchs, "Spatially Augmented Reality," *Augmented Reality: Placing Artificial Objects in Real Scenes* (Proc. First IEEE Workshop Augmented Reality, IWAR 98), R. Behringer, G. Klinker, and D. Mizell, eds., A.K. Peters, Natick, Mass., 1998, pp. 64-71.
2. M. Walker, *Ghostmasters: A Look Back at America's Midnight Spook Shows*, Cool Hand Communications, Boca Raton, Fla., 1994.
3. K.C. Knowlton, "Computer Displays Optically Superimpose on Input Devices," *Bell Systems Tech. J.*, vol. 53, no. 3, March 1977, pp. 36-383.
4. T. Poston and L. Serra, "The Virtual Workbench: Dextrous VR," *Proc. Virtual Reality Software and Technology* (VRST 94), IEEE CS Press, Los Alamitos, Calif., 1994, pp. 111-121.
5. T.E. Weigand, D.W. von Schloerb, and W.L. Sachtler, "Virtual Workbench: Near-Field Virtual Environment System with Applications," *Presence*, vol. 8, no. 5, Oct. 1999, pp. 492-519.
6. C. Chinnock, "Holographic 3D images Float in Free Space," *Laser Focus World*, vol. 31, no. 6, June 1995, pp. 22-24.
7. S. McKay et al., "Membrane Mirror Based Display For Viewing 2D and 3D Images," *Proc. SPIE 3634—Projection Displays V*, vol. 3634, SPIE Press, Bellingham, Wash., 1999, pp. 144-155.
8. H. Fuchs et al., "Adding a True 3-D Display to a Raster Graphics System," *IEEE Computer Graphics and Applications*, vol. 2, no. 7, Sept. 1982, pp. 73-78.
9. O. Bimber, L.M. Encarnação, and D. Schmalstieg, "Augmented Reality with Back-Projection Systems using Transflective Surfaces," *Computer Graphics Forum* (Proc. Eurographics 2000), vol. 19, no. 3, Blackwell Publishers, Oxford, UK, 2000, pp. 161-168.
10. H. Löffelmann and E. Gröller, "Ray Tracing with Extended Cameras," *J. Visualization and Computer Animation*, vol. 7, no. 4, Oct. 1996, pp. 211-228.
11. M. Segal et al., "Fast Shadows and Lighting Effects Using Texture Mapping," *Computer Graphics* (Proc. Siggraph 92), ACM Press, New York, 1992, pp. 249-252.
12. T. Möller and B. Trumbore, "Fast, Minimum Storage Ray-Triangle Intersection," *J. Graphics Tools*, vol. 2, no. 1, Jan. 1997, pp. 21-28.

Oliver Bimber *is a scientist at the Fraunhofer Institute for Computer Graphics and is a PhD candidate at the Darmstadt University of Technology, Germany. His research interests are in display technologies, rendering, and human–computer interaction for mixed realities. He received a BS in commercial computing from the Dundalk Institute of Technology, Ireland and a Dipl.Inform (FH) in scientific computing from the University of Applied Science Giessen, Germany.*

Bernd Fröhlich *is a professor for Virtual Reality Systems with the media faculty at the Bauhaus University Weimar in Germany. His recent work focuses on input devices, interaction techniques, display systems, and support for tight collaboration in local and distributed virtual environments. He received his MS and PhD in computer science in 1988 and 1992, respectively, from the Technical University of Braunschweig, Germany.*

Dieter Schmalstieg *is an assistant professor at the Interactive Media Systems group at Vienna University of Technology, Austria, where he directs the Studierstube augmented reality research project. His research interests include virtual environments, augmented reality, 3D user interfaces, and real-time graphics. He holds a PhD and Habilitation degree from Vienna University of Technology.*

L. Miguel Encarnação *is the head of the Human Media Technologies Department of the Fraunhofer Center for Research in Computer Graphics in Providence, Rhode Island, and is an adjunct professor in computer science at the University of Rhode Island. His research interests include next-generation user interfaces, mixed reality technologies, and advanced distributive learning environments. He received his PhD in computer science in 1997 from the University of Tübingen, Germany.*

Readers may contact Bimber at Fraunhofer Center for Research in Computer Graphics (CRCG), 321 S. Main St., Providence, RI 02903, email obimber@crcg.edu.