Journal of Zhejiang University SCIENCE A
ISSN 1673-565X (Print); ISSN 1862-1775 (Online)
www.zju.edu.cn/jzus; www.springerlink.com
E-mail: jzus@zju.edu.cn

Using relief texture for interactive and tangible virtual environments*

Junae KIM[1], Seonhyung SHIN[1], Gerard Jounghyun KIM[†‡2]

([1]*Digital Contents Division, Electronic and Telecommunications Research Institute, Daejon 305-700, Korea*)

([2]*Department of Computer Science and Engineering, Korea University, Seoul 136-713, Korea*)

[†]E-mail: gjkim@korea.ac.kr

Received Dec. 10, 2007; revision accepted May 4, 2008; CrossCheck deposited Nov. 10, 2008

Abstract: This paper first introduces a way to improve interactivity with high polygon count virtual objects through the “mixed” use of image-based representation within one object. That is, both 3D polygonal and image-based representations are maintained for an object, and switched for rendering depending on the functional requirement of the object. Furthermore, in order to reduce the popping effect and provide smooth and gradual transition during the object representation switch, the object is subdivided with the subdivided parts possibly represented differently, i.e., using 3D models or images. As for the image-based representation, the relief texture (RT) method is used. In particular, through the use of the mixed representation, a new way called TangibleScreen is proposed to provide object tangibility by associating the image-based representation with a physical prop (projecting the RTs) in a selective and flexible way. Overall, the proposed method provides a way to maintain an interactive frame rate with selective perceptual details in a large-scale virtual environment, while allowing the user to interact with virtual objects in a tangible way.

Key words: Relief texture (RT), Image-based modeling, Tangible interaction, Virtual environment
doi:10.1631/jzus.A0720113 **Document code:** A **CLC number:** TP391

INTRODUCTION

A virtual environment is organized as a collection of virtual objects represented and rendered with 3D polygonal models. Image-based objects (or image-based rendering techniques in general) have been suggested as a probable alternative to 3D polygonal models for their advantage in providing a constant frame rate and photo-realism (Chen and Williams, 1993; Aliaga and Lastra, 1999; McMillan, 2001). That is, the computation or graphic processing time required to render image-based objects is proportional largely only to the size and number of the images (which remains more or less constant) being manipulated to generate the correct view. However, image-based techniques are not suited for close range and direct interaction due to depth inconsistency and the lack of ability to represent dynamic behaviors.

For instance, in an application such as a virtual museum, virtual artifacts usually require reasonably high realism and interaction with them, and even some behavior. Thus, the object may contain a very high number of polygons (in the range of tens of thousands) to express all its details. When several objects (including the interaction object) are visible at once, it may not be possible to achieve interactive frame rates (e.g., 30 frames/s) on a desktop PC. One possible solution is to maintain both image-based representations and 3D polygonal models and switch between them, that is, to maintain the right mix of respective representations among the objects for interactive rendering of the scene. However, such an approach may introduce delay, break, or popping effects in switching between the image-based representations and the 3D polygonal models due to, for example, the overhead in loading a high number of

[‡] Corresponding author

[*] Project (No. R01-2006-000-11142-0) supported by the “Teukjung Gicho” Program of the Korea Science Foundation

polygons at once. Such a popping effect and large variation in the frame rate is known to degrade the presence and the quality of the virtual experience (Reddy, 1997; Luebke *et al*., 2002).

In this paper, we first propose two representations—the relief texture (RT), which is an image-based object technique developed by (Oliveira and Bishop, 1999; Oliveira *et al*., 2000), and the conventional 3D polygons—to represent a virtual object. We choose the RT for its efficient and fast implementation that takes advantage of texture mapping hardware available in today's graphics board. We further propose to subdivide an object and provide the dual representation for each subdivision. Thus, a virtual object is represented in part by images and in part by 3D polygons. We call this approach the "mixed" representation. This allows a gradual switch between the image-based and 3D polygonal representations and has other advantages (leveraging on merits of the respective representations) in providing better visibility, behavior representation, and improved interactivity (e.g., by using a polygonal representation in close range and image-based representation for far objects).

Furthermore, building on the characteristics of RT, which paste the warped textures on a six-sided polyhedron, we propose a way to interact with the object in a tangible way by projecting the RTs of the interaction object on a small physical prop (cubic or pyramidal). The prop screen, named the "TangibleScreen", is general since many different objects can be projected on it as needed. Such tangibility and feel for concreteness can much improve the virtual experience (Ishii and Ullmer, 1997).

In the next section we first survey previous works on image-based techniques, other attempts of the "hybrid" representation of virtual/3D environments, and related works on object centric projection and tangible interfaces. We also give a short review of the RT and discuss its limitations in providing complete visibility, and interactive behaviors. Section 3 gives an overview of the proposed system, and in Section 4 we give a detailed explanation to the mixed representation approach. In Section 5, we describe how we apply the mixed representation to tangible interaction, by projecting the RT on the TangibleScreen. Finally, we conclude the paper with a summary and plans for future work.

RELATED WORKS

Hybrid representation and image-based objects

Images and 3D geometric models have been used together in a virtual environment, separately for partial scenes and predetermined objects. Schaufler (1995; 1997) proposed the concept of the nailboard and imposter, the dynamically created textures to replace distant 3D objects or objects beyond portals in real time. Such techniques are used to maintain and regulate interactive system performance (Aliaga and Lastra, 1998; Jeschke *et al*., 2005). However, these approaches generally lack abilities to represent interactive behaviors. Other directions to regulating system performance for large-scale virtual environments include the traditional level-of-detail (LOD) methods and out-of-core rendering (Luebke *et al*., 2002; Oat *et al*., 2006).

The image-based object approach (Oliveira and Bishop, 1999) models the entire individual object (as opposed to a partial view) using image-based rendering techniques. Early image-based rendering researchers have proposed to regularly sample object images 360° around the object, and to generate an arbitrary view by interpolating nearby views with the closest viewpoints (Chen and Williams, 1993; Seitz and Dyer, 1996; McMillan, 2001). These approaches generally require a large number of images. Oliveira *et al*.(2000) proposed an approach called RT that uses only six images around an object and efficiently warps them according to a given viewpoint. Since then, RTs have been extended in many ways including efficient GPU-based computation and the application to a larger class of complex-shaped objects (Fujita and Kanai, 2002; Baboud and Decoret, 2006; Andujar *et al*., 2007).

We have adopted and refined the RT as originally proposed by Oliveira *et al*.(2000) for our "mixed" representation that employs both images and 3D polygons "within" a single object. The RT mapping represents an object by a six-sided textured bounding box of the object (Fig.1). In fact, the object is created as a six-sided layered depth image (Shade *et al*., 1998), whose texture and depth information are derived by an orthogonal projection of the 3D model onto the six surfaces. The object textures are warped according to the changing viewpoint. Oliveira *et al*. (2000) have formulated a way to efficiently compute the newly warped texture coordinate from the reference pixel, using the corresponding 3D point, the reference viewpoint, and the new viewpoint.

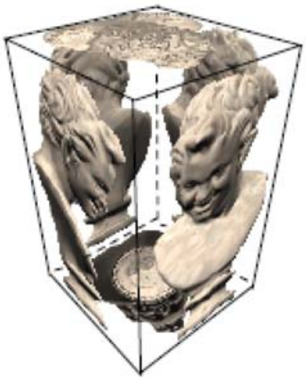


Fig.1 With relief textures, an object is modeled using a six-sided textured box with depth information (Oliveira *et al*., 2000)

Projection on non-parallel surfaces

Raskar *et al*.(1998) have considered the problem of projecting background scenes for stage design purposes on hanging curved drapes, and proposed a method of pre-distortion to generate the correct view. However, their work assumed a fixed viewpoint and required a large amount of computation, which are not appropriate for real-time interactive applications. Raskar *et al*.(1998) used the method of the projective texture mapping and a two-stage rendering method to project images on irregularly shaped surfaces. It took advantages of the built-in capabilities of modern graphics hardware and could run in real time. The use of projective texture involves moving the rendered image in the frame buffer to the texture memory, and texture-mapping it to a virtual screen. While the texture mapping stage can use the built-in graphics hardware support, it still requires the initial rendering directly dependent on the number of polygons of the objects or the scene. On the other hand, RTs can be used in a similar way to project an object image on six-sided prop object surfaces. The difference with using projective textures would be that, instead of the initial rendering of the polygons, one must warp the six textures depending on the user's viewpoint. The projection parameters are exactly the same as those in the texture mapping on the virtual six-sided bounding box. In this way, the rendering process takes constant time, regardless of the object complexity. Projection of images on physical objects has been used in the context of augmented reality for giving tangibility to virtual objects (Bimber, 2002). Raskar and Low (2001) also used the method described above to project graphics images onto non-moving physical objects with the same corresponding shapes but without the internal details (color, decal, etc.). In a similar context, Low *et al*.(2001) and Pair *et al*.(2003) applied image projections on life sized objects to create large-scale mixed/augmented reality environments. Similar to our work, Grossberg *et al*.(2004) used a projector and a camera system to make one object look like another by projecting a carefully determined compensation image onto the object.

SYSTEM OVERVIEW

The vision of the proposed virtual environment is illustrated in Fig.2. Each object maintains image and 3D polygonal representations for its subdivisions and the appropriate representation would be used to render them (e.g., distant objects mostly rendered with images, and close objects or objects with behaviors mostly rendered with polygons). Note that the different resolutions of the image representation can also be maintained to further regulate performance. Thus, the interaction object can be selected to be projected onto a prop with high-resolution image RTs. A good balance of system performance, interactivity/tangibility, and image quality can be maintained.

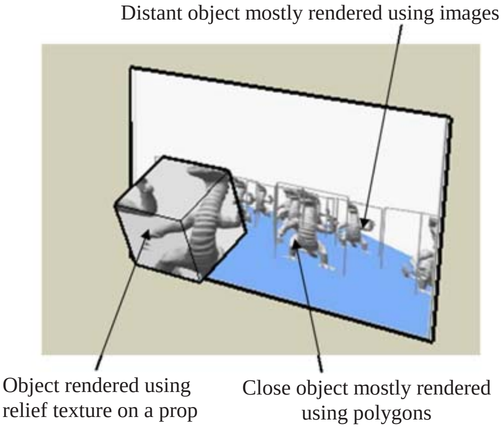

Distant object mostly rendered using images
Object rendered using relief texture on a prop
Close object mostly rendered using polygons

Fig.2 Vision of the proposed system. Virtual objects and their subparts are rendered using mixtures of images and polygons for performance optimization, and one selected object is projected onto a prop for closer tangible inspection

MIXED REPRESENTATION WITHIN AN OBJECT

To tackle the situation in which individual virtual objects are made of a high number of polygons, we propose to use both image- and polygon-based representations and switch between them appropriately. In addition, to address the problem in just using the image-based approach, or using a naive hybrid approach, we propose to subdivide the object and "mix" different representations for different parts of the object. That is, we maintain a dual representation for each subdivision (In this work, the subdivision of the object is manually done), and select the appropriate representation according to some criteria like minimization of the popping effect, invocation of interactive behaviors, maintenance of the minimum user perception level and the frame rate. We call this approach the "mixed" representation (Fig.3). The next subsections will first describe the basic components of the mixed representation, namely, the data structure support, the criteria used for manually subdividing a given object, the rendering method, and the method and criteria for the representation switch (at the subdivision object level). We also explain how to represent object behaviors using the mixed representation, and address the issue of building large virtual environments using mixed representation objects.

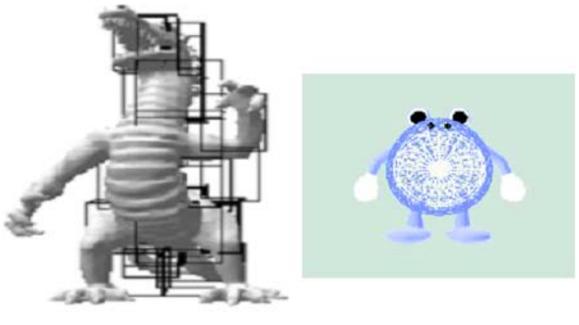

(a) (b)

Fig.3 Mixed representation of a virtual object. An object is subdivided and each subdivision is represented using 3D polygons or relief textures (RTs). (a) In this dinosaur model (about 10 000 polygons), the front part is rendered with 3D polygons, and the rear (enclosed with boxes) rendered using the RTs; (b) Pokemon's stomach (rendered with a wire frame for illustration) is composed of polygons while others rendered using the RT

Mixed object node: the data structure

In order to support the mixed representation, we have implemented an integrated scene graph node in OpenGL (Neider, 1993). The data structure resembles an octree, in which parent nodes are further subdivided into children nodes in a recursive fashion. The subdivision is to be carried out manually using conventional geometric modeling tools (criteria for the subdivision are given in the next subsection). Each intermediate node represents a particular subdivision and contains the relevant 3D polygons, six orthogonally projected textures (i.e., RTs), pixel by pixel depth values, the normal vector of the six faces, and other miscellaneous information. The textures and depth information are easily acquired from the 3D synthetic object by an orthogonal projection onto the sides of its bounding box (we do not consider representing real objects in this study). We have ignored the redundantly represented information across the subdivision boundaries for now. That is, polygons may belong to more than two subdivisions; however, this does not affect the workings of the overall approach in principle and practically would not affect the overall performance in a significant way, since the portion of the overlapped subdivision is assumed to be not much compared to the total number of polygons.

Subdivision criteria

The criteria for manually subdividing an object are as follows. In order to address the visibility and hole problem, subdivisions were made for parts of the object with substantial concavity. In this way, holes can be avoided as much as possible, when the object (or the particular subdivision cell) is seen from arbitrary angles with the image representation. The second criterion considered was behavior and sub-object independence. For instance, if a motion behavior like "rotating an arm" or "moving a finger" is expected of the object, the corresponding part would be designated as a subdivision (Fig.4). Finally, we considered the number of polygons contained in one subdivision. It would be desirable for the number of 3D polygons contained in a subdivision to be kept within a certain limit for preventing the popping effect in switching the representation.

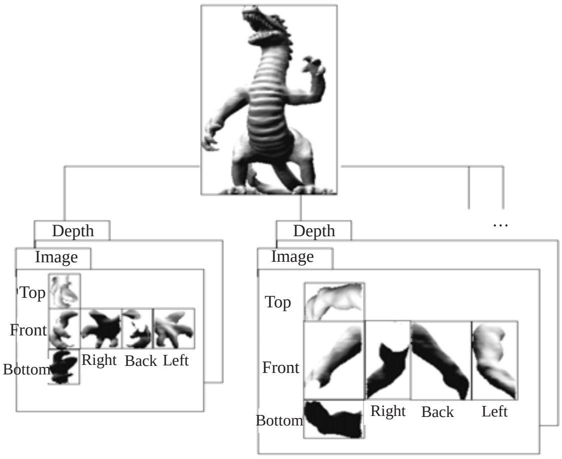


Fig.4 Subdivision data structure with the mixed representation

Rendering algorithm

The rendering algorithm for the "subdivided" RT is exactly the same as the original one. The only difference is that we repeat the rendering process for each subdivision in a back-to-front order. Note that we only need to consider at most three textures per subdivision (i.e., only three surfaces are visible at one time). Fig.5 illustrates the concept. In Fig.5, for example, the three visible surfaces in the rear region would be rendered first, and then the three surfaces in the front (The order among the three does not matter).

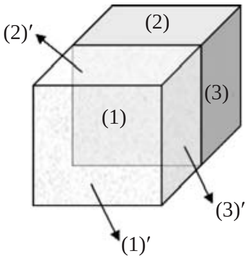


Fig.5 Back-to-front ordering for rendering the relief textures with subdivided objects. The order is (1)→(2)→(3), then (1)′→(2)′→(3)′. The order among (1), (2), (3), or (1)′, (2)′, (3)′ is unimportant

Switching between representations

Ideally, the switching for subdivision should be based on the human visual or depth perception. For instance, for a given subdivision, we switch from the image representation to the 3D polygon model (and vice versa), when each pixel of the texture projects to more than one pixel on the viewing plane to avoid the aliasing effect (we assume that the 3D model of the object is highly detailed). However, maintaining a constant and acceptable frame rate (e.g., 10~15 frames) is equally important in real-time virtual environments. Given positions P_A and P_O, of a user A and an object O, and the angle between the view direction and the direction toward the object, θ_{AO}, the relative importance $I(A, O)$ of the object is formulated as follows:

$$I(A,O) = u \cdot P(O) + \frac{v}{\|P_A - P_O\|} + w \cdot \cos\theta_{AO} + x \cdot S(O),$$
$$0 \le u,\ v,\ w,\ x \le 1, \tag{1}$$

where $P(O)$ is the number of pixels on the image plane that corresponds to one texture pixel, and $S(O)$ is the screen space area of the virtual object. The values of the weights u, v, w, and x, are determined empirically; they can be adjusted to emphasize certain aspects, e.g., performance or visual quality.

Object behaviors

One of the goals of using the mixed representation is to allow a more flexible implementation of object behaviors. One straightforward way to achieve this is to have the object representation switched to the 3D geometry and to apply 3D transformation or vertex moving simulation (in the case of motion behavior or deformation). This is still problematic as the geometry changes (by the object behavior) occur within a subdivision. In such a case, when the object switches back to an image representation, its RTs must be changed according to the result of the behavior. A careful subdivision would be needed at the modeling stage to ensure that the desired behaviors can be represented by moving the object at the subdivision level. Thus, with this restriction, if the lighting effect can be ignored to some degree, we can simply apply 3D transformation or simple behavioral effects (e.g., change the color) directly to the image representation (i.e., RT box) instead (Fig.6).

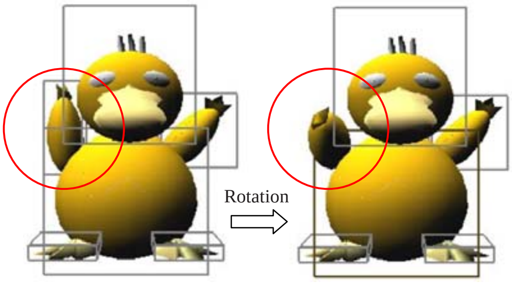


Fig.6 Pokemon's right arm is rotated as a relief texture box

Building a virtual environment with mixed representation objects

This subsection discusses issues in incorporating multiple virtual objects with the mixed representation into a virtual environment. Normally, a graphics hardware includes a culling module that traverses the scene graph and hands 3D polygons to be processed to the next rendering module. However, as for relief textured objects (or subdivisions), the warping computation must be carried out before the scene graph enters the graphics pipeline. Thus, objects (or subdivisions) must be culled by software from the view frustum (a visible volume whose parameters are given by the user).

The virtual objects may move around and their bounding boxes may collide with one another, even though the object themselves are not in collision. In this case, if we treat the objects separately, a correct view may not be generated. Fig.7 illustrates the situation. If Object 1 is rendered ahead of Object 2, the part that should be occluded (at the middle part of the figure) will not come out right. Therefore, when the bounding boxes collide, we merge the two subdivision cells into a virtual bounding box to reduce the number of texture operations.

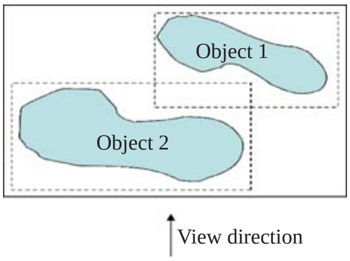
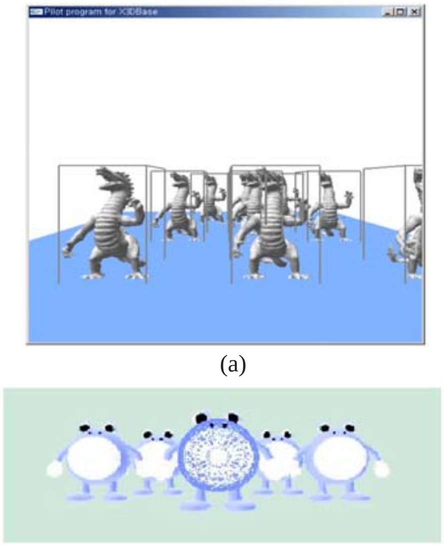


Fig.7 Two different subdivision cells belonging to two different objects in collision. Object 1 occludes Object 2 in the middle when seen in the arrow direction. The two cells are merged to form a bigger relief texture object (with the bigger bounding box drawn in solid lines)

Fig.8 shows a simple virtual environment with many virtual objects with the mixed representation. Although indistinguishable visually, the mixture of the representations among all subdivision cells is determined by placing a constraint on the number of polygons that the given system can handle per second (in this case, 5000). Within this constraint, we apply Eq.(1) at every frame to decide which objects (or cells) are important enough to be rendered in 3D polygons. Fig.9 shows the variation in the frame rate as a user moves around the scene. Note that the frame rate never drops below 10 frames/s (the variation above the minimum frame rate is much less problematic).



(a)

(b)

Fig.8 A virtual environment with multiple virtual objects with mixed representation. (a) Dinosaur model; (b) Pokemon model

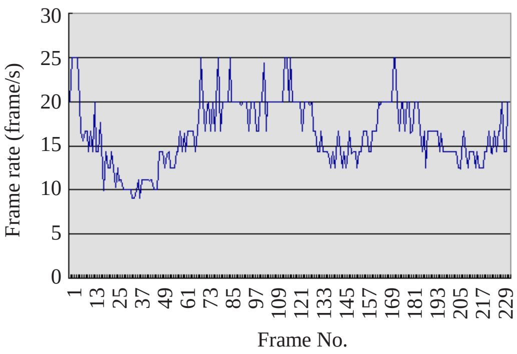


Fig.9 Frame rate trend when a user navigates in the virtual environment of Fig.7. A dynamic switching of representations is applied to keep the target frame rate above 10 frames/s

TANGIBLESCREEN: TANGIBILITY USING MIXED REPRESENTATION

In addition to the several advantages of the mixed representation objects described in Section 4, maintaining dual representations can further be used to create interaction tangibility and concreteness in a dynamic and flexible way. The idea is inspired by the nature of how the RT works, i.e., “projection on the bounding box”. We take this metaphor to the real world, and use a six-sided prop object as a physical representative of a virtual object overlaid on it (we call it the “TangibleScreen”). Even though the actual shape of the virtual object is not the same as the prop screen, our belief is that it still gives the feeling of tangibility, concreteness, and depth perception to enhance the presence of the object in the corresponding virtual space. The user can physically hold, manipulate, and examine the virtual object overlaid on the six-sided prop.

In Section 4, our general approach was to switch back to the polygonal representation for the virtual objects when they were near the user, for instance, during the interaction. However, in order to apply the “bounding box to prop” metaphor, the interaction object (i.e., the object projected on the prop) must be represented with RTs only. One of the emerging problems in the case of the TangibleScreen is how to make an appropriate projection on the moving screen. As the user holds and moves the prop in his/her hand to view and manipulate it, the projection images must be corrected. The prop must be tracked and registered with the virtual environment that it belongs to.

TangibleScreen

Fig.10 illustrates the idea of the TangibleScreen. When the TangibleScreen is within the projector projection volume (i.e., the pyramidal volume formed by the projection screen and the four edges emanating from the projection point) and the user’s view direction is within a tolerable difference with the projection direction, the user will be able to view the object through the TangibleScreen. The image on the TangibleScreen must be viewpoint dependent, thus both the TangibleScreen and the user’s view must be tracked. In our implementation, they were tracked by using two Polhemus 6-DOF trackers. The size of the TangibleScreen (the box) was determined at 12 cm× 12 cm×12 cm, considering that the size of the box (minimum of the width, length, and height) must be at least larger than the inter-ocular distance (~6.5 cm), otherwise the human user could see more than three sides of the box at the same time, and that the box has to be small enough to be held by a user (Fig.10).

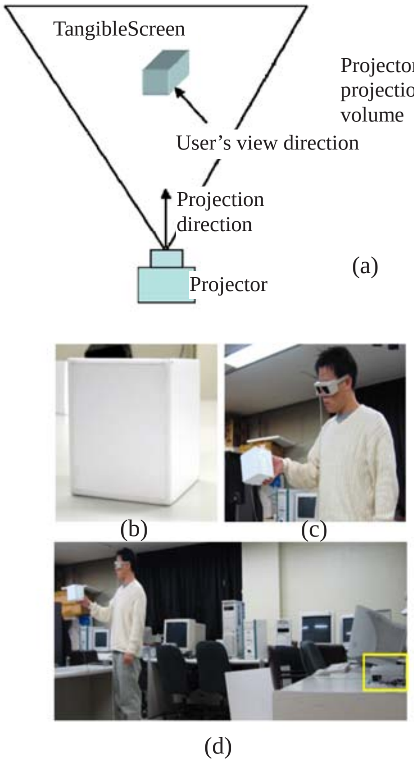


Fig.10 Overview of the TangibleScreen. (a) Relief textures are projected on a hand-held six-sided prop; (b) and (c): to create the feeling of tangibility to a virtual object; the yellow inset in (d) indicates the location of the projector

In order to produce the feeling of mixed reality, that is, a seamless connection between the TangibleScreen virtual object and the virtual space displayed through the larger projection screen behind, stereoscopy is used (naturally, the object mapped on the TangibleScreen is in the negative parallax zone, with respect to the user). Since the virtual object mapped on the TangibleScreen belongs to the larger virtual space displayed along with it, coordinates of the real space and the virtual space must be registered, just like in any mixed-reality system. A simple manual calibration process was employed as follows:

Step 1: Measure (in real world units) four corners of the projection image rectangle at two projection distances. The coordinate of the center of projection is at the intersection point (approximated) of four lines, each formed by extending the corresponding corners. The horizontal and vertical projection angles are easily obtained in this way.

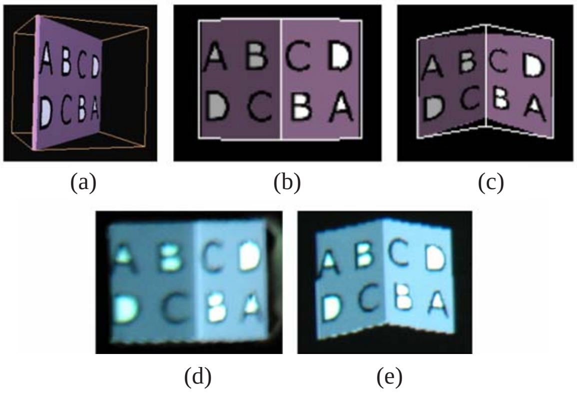
Step 2: Place a virtual RT object at various distances in the virtual/tracker space. For each placement, manually find the corresponding real space positions by translating the TangibleScreen of size *s* closer to the projector from a sufficiently far distance until the textures are projected and focused on the TangibleScreen (call it distance, *d*).

Step 3: Use the correspondences between the physical measurements (e.g., box size *s* and distance *d*) and the virtual coordinates of the virtual RT object, to calibrate the units between the virtual and the real space. Calibrate the image plane and the center of projection (measured in real space in Step 1) to match that in the virtual/tracker space. The camera (projection) parameters for rendering are set equal to these values.

In order to implement the TangibleScreen that moves within the projection volume, it is necessary that the projector likewise be able to automatically produce a reasonable focus at different positions with respect to a fixed center of projection. With the CRT (cathode ray tube) based projectors, this is difficult to achieve as there are three separate CRTs for the three principal axes with separate optical systems. This mechanical structure makes the CRT based projectors very sensitive in terms of focusing the image at different distances (manual adjustment will be needed every time the projection distance is changed). DLP (digital light processing) projectors, on the other hand, have a different mechanical structure (e.g., single lens system) that produces a reasonable focus at any point along the line of projection with higher image resolution and quality. Thus, we chose to use the DLP projector to implement the TangibleScreen.

Rendering

The RTs of the interaction object were warped according to the user's view direction and then projected (texture mapped) on a virtual cube, whose position coincided with the physical counterpart. Fig.11 illustrates several cases of image projections. Fig.11a simply illustrates a planar object (the object to be rendered using the RTs) with respect to the virtual bounding box. The original textures were warped and pasted on the bounding box and seen from the user's viewpoint (Fig.11b). The same textured cube was seen then from the projector (Fig.11c). Since the textures were warped with the user's viewpoint in consideration, the textures naturally do not align with the bounding box when seen from this viewpoint. Fig.11d shows the actual projection but on a flat surface and Fig.11e shows the actual projection on the physical cube as seen from the user. Note that the textures were correctly projected in the latter case.


(a) (b) (c)

(d) (e)

Fig.11 Several cases of image projections. (a) A planar object (the object to be rendered using the relief textures) situated with respect to the virtual bounding box; (b) Original textures warped and pasted on the bounding box and seen from the user's viewpoint; (c) The same textured cube seen from the projector (Since the textures were warped with the user's viewpoint in consideration, the textures naturally do not align with the bounding box when seen from this viewpoint); (d) Actual projection but on a flat surface; (e) Actual projection on the physical cube as seen from the user (Note that the textures are correctly projected in this case)

Guiding the user

The TangibleScreen is "viewable" only if the surfaces (among the six sides of the prop box) visible to the user are also visible (or open to projection) to the projector. In order for this situation not to occur, the user must be constrained in a small restricted zone. With a fixed projector location, there is no way to overcome this problem. Instead, we propose to give the user guidelines (through synthesized voice, for instance) so that the user can move or manipulate the TangibleScreen and the visible surfaces of the user and the projector coincide.

We test the visibility (by the angle difference between the view/projection directions and the normals of the surfaces) of the six sides of the TangibleScreen from the projector and the user's point of view. If the two sets of the visible surfaces do not coincide (we call this situation "invalid", and "valid" if coincident), the system will try to generate an advice (e.g., by voice or text messages) to the user so that the user can take actions to make the two sets of visible surfaces as coincident (valid) as possible. The potential for the greatest amount of change toward making the situation more valid is derived by computing the screen space area of the surface which is visible to the user but on which no image is projected. An advice is given to the user to take an action (moving or rotating the TangibleScreen) to make this surface open to image projection while keeping the originally valid set unchanged. Note that there are at the maximum three surfaces that may be visible to the user but no images are projected on. Fig.12 shows a case where there is no image projected on the right side of the TangibleScreen (due to the user's view direction). By rotating the TangibleScreen counterclockwise, the situation becomes valid.

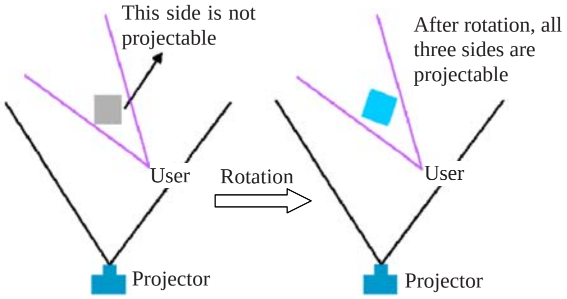

This side is not projectable
User
Rotation
Projector
After rotation, all three sides are projectable
User
Projector

Fig.12 Asking the user to rotate the TangibleScreen to make its right side open to image projection so that the user can see it

FrustumScreen

Having the user adjust the pose of the TangibleScreen to receive full image projection is of great nuisance. One way to effectively use the TangibleScreen is to limit the degrees of freedom of the user with respect to the image projection. Thus we propose a special configuration where the TangibleScreen is shaped in a pyramidal frustum shape with the projector located right above it, and restrict the movement to the TangibleScreen on the 2D area of the table (Fig.13). By this way, the TangibleScreen always receives projection on all of its sides (except for the bottom which is not needed). We call this particular system setup the FrustumScreen.

Implementation results

Fig.14 shows a vase projected on the FrustumScreen as part of a virtual museum application. In this application, being able to manipulate artifacts with tangibility, in addition to just viewing them, would be of great importance. The user can select an object and a viewing surface on the FrustumScreen and manipulate it (move it around on the 2D table surface). One can easily see the difference from the other objects that are displayed flat in terms of the 3D look and concreteness. The application was run in a dark environment to minimize visible difference between the virtual object and the FrustumScreen itself.

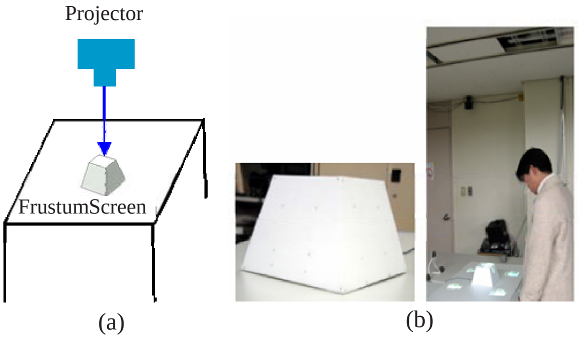


Fig.13 (a) FrustumScreen on the table; (b) The actual FrustumScreen and a user looking at a projection on the FrustumScreen

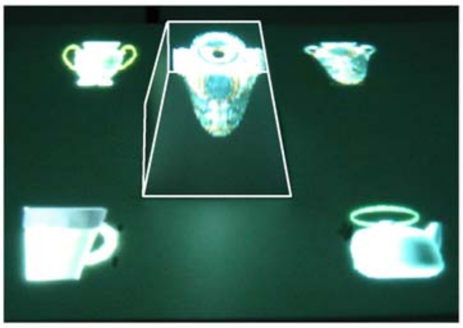


Fig.14 A vase projected on the FrustumScreen as part of the virtual museum application

CONCLUSION

In this paper, we presented an approach to mixing both polygonal- and image-based representations (RTs) within an object in rendering a virtual object. The main motivation is in enabling real-time rendering (with minimal frame rate variation) for virtual objects with high polygon counts while providing the minimum level of perceptual details. The mixed representation helps reduce the popping effect by gradually switching the representations subdivision by subdivision as necessary. In particular the mixed representation extends easily to implementing interactive object behaviors and tangible interaction, not possible with conventional virtual environments. Several future directions remain to further improve our framework. One is an incorporation of the GPU computation for the RTs (Fujita and Kanai, 2002; Baboud and Decoret, 2006) to make our framework faster and to process a much larger number of objects. Another direction is to investigate a way to make the object subdivision process (semi)automatic to make the modeling process easier for the user.

References

Aliaga, D., Lastra, A., 1998. Architectural Walkthroughs Using Portal Textures. Proc. Virtual Reality Annual Int. Symp., p.228-233.

Aliaga, D., Lastra, A., 1999. Automatic Image Placement to Provide a Guaranteed Frame Rate. Proc. ACM SIGGRAPH, p.307-316.

Andujar, C., Boo, J., Brunet, P., Fairen, M., Navazo, I., Vazquez, P., Vinacua A., 2007. Omni-directional Relief Impostors. Proc. EUROGRAPHICS, **6**(3):553-560.

Baboud, L., Decoret, X., 2006. Rendering Geometry with Relief Textures. Proc. Graphics Interface, p.195-201.

Bimber, O., 2002. Interactive Rendering for Projection-based Augmented Reality Displays. PhD Thesis, Darmstadt University of Technology, Germany.

Chen, S., Williams, L., 1993. View Interpolation for Image Synthesis. Proc. ACM SIGGRAPH, p.279-288.

Fujita, M., Kanai, T., 2002. Hardware-assisted Relief Texture Mapping. Proc. EUROGRAPHICS, p.257-262.

Grossberg, M., Peri, H., Nayar, S., Belhumeur, P., 2004. Making One Object Look like Another: Controlling Appearance Using a Projector-camera System. Proc. Conf. on Computer Vision and Pattern Recognition, p.452-459.

Ishii, H., Ullmer, B., 1997. Tangible Bits: Towards Seamless Interfaces Between People, Bits and Atoms. Proc. SIGCHI Conf. on Human Factors in Computing Systems, p.234-241. [doi:10.1145/258549.258715]

Jeschke, S., Wimmer, M., Purgathofer, W., 2005. Star: Image-based Representations for Accelerated Rendering of Complex Scenes. Proc. EUROGRAPHICS, p.1-20.

Low, K., Welch, G., Lastra, A., Fuchs, H., 2001. Life-sized Projector-based Dioramas. Proc. ACM Symp. on Virtual Reality Software and Technology, p.93-101. [doi:10.1145/505008.505026]

Luebke, D., Reddy, M., Cohen, J., Varshney, Watson, B., Huebner, B., 2002. A. Level of Detail for 3D Graphics. Morgan Kaufmann, USA.

McMillan, L.Jr., 2001. An Image-based Approach to Three-dimensional Computer Graphics. PhD Thesis, University of North Carolina at Chapel Hill, USA.

Neider, J., 1993. OpenGL Programming Guide. Addison Wesley Publishing Company, USA.

Oat, C., Mitchel, J., Evans, A., Wenzel, C., 2006. Advanced Real-time Rendering in 3D Graphics and Games. Siggraph Course Notes 26.

Oliveira, M., Bishop, G., 1999. Image-based Objects. Proc. ACM Symp. on Interactive 3D Graphics, p.191-198. [doi:10.1145/300523.300552]

Oliveira, M., Bishop, G., McAllister, D., 2000. Relief Texture Mapping. Proc. ACM SIGGRAPH, p.359-368.

Pair, J., Neumann, U., Piepol, D., Swartout, B., 2003. FlatWorld: combining Hollywood set design techniques with VR. *IEEE Comput. Graph. Appl.*, **23**(1):12-15. [doi:10.1109/MCG.2003.1159607]

Raskar, R., Low, K., 2001. Interacting with Spatially Augmented Reality. Proc. AFRIGRAPH, p.101-108. [doi:10.1145/513867.513889]

Raskar, R., Welch, G., Cutts, M., Lake, A., Stesin, L., Fuchs, H., 1998. The Office of the Future: A Unified Approach to Image Based Modeling and Spatially Immersive Displays. Proc. ACM SIGGRAPH, p.179-188.

Reddy, M., 1997. The Effects of Low Frame Rate on a Measure for User Performance in Virtual Environments. Technical Report ECS-CSG-36-97. Department of Computer Science, University of Edinburgh, UK.

Schaufler, G., 1995. Dynamically Generated Imposters. Proc. Workshop on Modeling Virtual World—Distributed Graphics, p.129-136.

Schaufler, G., 1997. Nailboards: A Rendering Primitive for Image Caching in Dynamic Scenes. Proc. EUROGRAPHICS Workshop on Rendering, p.151-162.

Seitz, S., Dyer, C., 1996. View Morphing. Proc. ACM SIGGRAPH, p.21-30.

Shade, J., Gortler, S., He, L., Szeliski, R., 1998. Layered Depth Images. Proc. ACM SIGGRAPH, p.231-242.