ICAT 2002
December 4-6, Tokyo, JAPAN

Mixing 3D Polygons and Relief Textures for Virtual Objects

Junae Kim

Virtual Reality Research Center,
Computer and Software Technology Laboratory
Electronics and Telecommunication Research Institute (ETRI),
Daejon, Korea

Gerard J. Kim

Virtual Reality Laboaratory, Dept. of Computer Science and Engineering
Pohang University of Science and Technology (POSTECH),
Pohang, Korea
gkim@postech.ac.kr

Abstract

Most often, a virtual environment is organized as virtual objects represented and rendered with 3D polygonal models. Image-based objects have been suggested as a probable alternative representation for its advantage in providing constant/acceptable frame rate when the 3D representation contains too many polygons [7]. However, since it is difficult to provide interaction (a vital component in virtual environments) through image based objects, a natural extension is to keep both representations, 3D and images, and switch between them depending on the functional requirement of the object [4]. Still, this may introduce a significant popping effect when one representation is switched to the other. This paper thus proposes to subdivide an object, and mix the two representations within one object (at the cell level) so that a smooth and gradual transition can be made. As for the image based representation, we use the Relief Texture method proposed by [8]. The criteria for object subdivision are based on the object structure, visibility, and polygon counts (per cell). The proportion of representation mixture is determined dynamically by factors like the viewing distance, angle, total allowable polygon counts, object importance, etc. The application of Relief Textures on the subdivision structure is made efficient with only three texture operations per object. Overall, the proposed method provides a way to maintain an acceptable or constant frame rate with selective perceptual detail and interactivity in a virtual environment with multitude of virtual objects.

Key words: Image based Objects, Rendering, Relief Texture, Virtual Objects, Virtual Environments

1. Introduction

Most often, 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 technique in general) have recently been suggested as a probable alternative to 3D polygonal models for its advantage in providing constant frame rate [7][8]. 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. The added bonus of using image based objects is the photorealism, while providing a roughly equivalent realism would usually require too many polygons to be rendered in real time (especially if there are many visible objects at one time). Rendering one or more image based objects may be achieved in real time depending on the required quality (e.g. image or sampling resolution), the specific algorithm, overall number of objects, etc. However, image based techniques (like the image based objects) are not suited for close range or direct interaction (due to depth inconsistency) or behavior representation (e.g. motion).

In an application like a virtual museum, for instance, virtual artifacts usually require reasonably high realism and interaction with them, and even some behavior. An interactive frame rate may not be possible when rendering all the visible objects due to the high number of polygons (on a desktop PC). A possible approach is to keep both image based representations and 3D polygonal models and switch between them (for actual rendering) depending on, for instance, their distances

from the user, while maintaining the right mix of respective representations among the objects for interactive rendering of the scene.

One problem still remains: the delay, break, or popping effect in switching from the image based representations to the 3D polygonal models due to the overhead in loading a high number of 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 [10][11].

In this paper, we first propose to use the Relief Texture, an image based object technique developed by [7][8], and 3D polygons to represent a virtual object. We chose the Relief texture for its efficient and fast implementation that takes advantage of the texture mapping hardware available in the today's graphics board. We further propose to subdivide an object and provide the dual representation for each subdivision. Thus, a virtual object is in part represented using images and in part by 3D polygons (See Figure 4). We will call this approach, the "mixed" representation. This allows a gradual switch between the image based and 3D polygonal representations. The subdivision technique also has other advantages in providing better visibility, behavior representation and improved interactivity.

In the next section we survey previous work in image based techniques (especially, image based objects) and other attempts in the "hybrid" representation. We also give a short review of the Relief Texture [8], its limitations in providing complete visibility, and interactive behavior. In Section 3, we give an overview of our approach in subdividing the virtual object and applying the mixed representation. It also covers more detailed topics relevant to the design of a single virtual object such as the criteria for subdivision, the data structure and rendering algorithm, and image based behavior. The next section talks about the design of a virtual environment using the virtual objects with the mixed representation. Finally, we conclude the paper with a summary and plans for future work.

2. Previous Work

2.1. Hybrid Representation

Images and 3D geometric models have been used together in a virtual environment, separately for partial scenes or predetermined objects. This approach is usually referred to as the hybrid representation. A typical application of hybrid representation is in the treatment of the background objects, or objects beyond the portal as images [1][2][3][14]. These approaches generally lack object-level interactive behavior.

The image based object approach [7] models individual objects (as opposed to a partial scene) using image based rendering techniques. Others proposed to regularly sample object images 360 degrees around the object, and to generate an arbitrary viewpoint, either by simply retrieving the image that represented the closest viewpoint or interpolating among few images with the closest viewpoints [12][15][16][17]. These approaches generally require a large number of images. Schaufler proposed the concept of the Nailboard and Imposter that replaced a far away 3D object with a texture in real time [5][6]. The texture would have to be regenerated intermittently as the user changes one's viewpoint. Yoon et al. has proposed to switch between image based and 3D representations at a closer distance from the user based on the human's capability to recognize depth within the object in a stereoscopic virtual environment [4]. Oliveira et al has proposed an approach called the Relief Texture that uses only six images around an object and efficiently warp them according to the viewpoint [8]. We adopt and refine the Relief Texture for our "mixed" representation that employs both images and 3D polygons within a single object.

2.2. Relief Texture

The two major approaches in the image based rendering approaches are the image warping [9] and the view interpolation [12][13]. In the warping approach, a new pixel value, $\bar{x}_2$, for a given new view point, $\dot{C}_2$, is computed using the pixel from the reference image, $\bar{x}_1$, taken from view point, $\dot{C}_1$, and the image projection matrices, P_1 and P_2. Below is the image warping equation given by McMillan [9] (note the equality is only up to scale).

$$\bar{x}_2 \doteq \delta(\bar{x}_1) P_2^{-1}(\dot{C}_1 - \dot{C}_2) + P_2^{-1} P_1 \bar{x}_1$$

This basic equation requires, for each input pixel, a total of 13 multiplications and 7 additions. On the other hand, a (correct) view interpolation method [13] requires a prewarp of two or more reference images to a reference direction, an interpolation between them, and a postwarp of the resultant image to the desired view point (e.g. the image plane is perpendicular to view direction toward the object/scene). The prewarping of reference images can be done off-line. These approaches were originally devised for rendering a scene rather than individual objects. The Relief Texture mapping works by, representing an object by a six sided textured bounding box of the object (See Figure 1). In fact, the object is created as a six sided Layered Depth Images [18], whose texture and depth information is derived by an orthogonal projection of the 3D model onto the six surfaces. The object textures are warped according to the changing view point.

Fig. 1 With Relief Textures, an object is modeled using a six sided textured box with depth information [8].

Consider applying the view interpolation technique to a scheme like above. We can try to generate the correct texture by warping and interpolating two or more reference images. However, an extra warping would be needed to account for the orientation of the final image plane (that contains multiple objects) whose optical axis would be different from the each view direction toward the individual object. Plus, a scaling operation is needed to account for the view distance as well (See Figure 2).

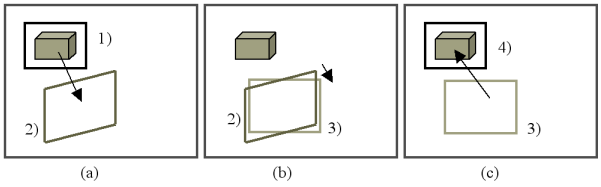

Fig. 2 Applying the view interpolation technique to Relief Texture. In order to compute the warped texture coordinates, an extra warping and scaling operation are needed.

Relief Texture technique has formulated a way to directly compute the warped texture coordinate, $\vec{x}_i$ from the reference pixel $\vec{x}_s$, using the corresponding 3D point, $\dot{x}$,the reference view point, $\dot{C}_s$, and the new view point $\dot{C}_t$. Here, for lack of space, we only show the final formula.

$$u_i = \frac{u_s + k_1 displ(u_s, v_s)}{1 + k_3 displ(u_s, v_s)}$$

$$v_i = \frac{v_s + k_2 displ(u_s, v_s)}{1 + k_3 displ(u_s, v_s)}$$

where (u_s , v_s) and (u_i , v_i)represent the source and target pixel, disp(u_s , v_s), the orthogonal displacement associated with the source pixel. k_1 , k_2 , and k_3 are constants for the given configuration of source and target cameras.

The detailed formulation is given in [8], and for each surface of the object considered, it only requires 6 multiplications and 4 additions, which is considerably less than the naive approach. Even though in a normal virtual environment, there would be many number of objects, and there would be (at most) three textures to consider for each object, this reduced amount of required computation at least gives some hope to using image based techniques for real time virtual environments.

There are some further considerations that need to be made to apply Relief Texture to real time virtual environments. The first is the visibility problem. Just

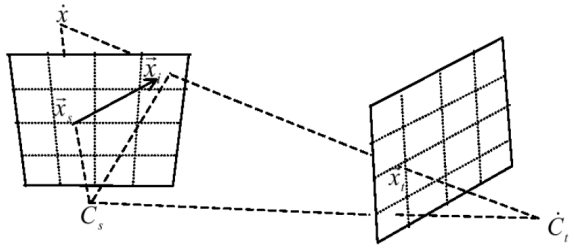

Fig. 3 Oliveira [8] has formulated a way to efficiently compute $\vec{x}_i$, the new texture coordinate when a new view point is given.

using six orthogonally projected textures (with depth information) can introduce many "holes" for objects with concavity. If the view point comes within the textured bounding box, the object will not be visible any more. This is usually not a problem unless the bounding box has large empty spaces (not tight). The aspect of interaction and behavior must also be considered, as these are very essential to making an effective virtual reality experience. It is generally difficult to express behavior, like motion, or implement some kind of 3D interaction when the object is represented in images.

3. Mixed Representation for a Single Virtual Object

To address the problem in just using the image based approach, like the Relief Texture, 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, and select the appropriate representation according to some criteria like invocation of interactive behavior, maintenance of minimum user perception level and frame rate. We call this approach, the "mixed" representation (See Figure 4).

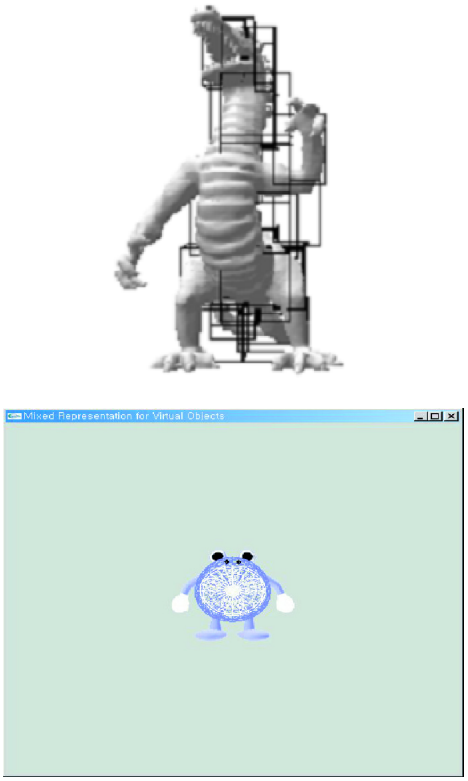

Fig. 4 Mixed representation of a virtual object. An object is subdivided and each subdivision is represented using 3D polygons or Relief Textures. (a) In this dinosaur model (the full 3D model has about 10,000 polygons), the front part is rendered with 3D polygons, and the rear parts enclosed with boxes are rendered using the Relief Textures. (b) The Pokemon's stomach (rendered with wireframe for illustration purpose) is composed of polygons while other parts of the body are rendered using the Relief Texture.

3.1. Mixed Object Node: The Data Structure

In order to support the mixed representation, we have implemented an integrated scene graph node in OpenGL [19]. The data structure resembles an octree, in which parent nodes are further subdivided into children nodes in a recursive fashion. Each intermediate node represents a particular subdivision and contains the relevant 3D polygons, six orthogonally projected textures, 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 paper). We have ignored the redundantly represented information across the subdivision boundaries for now

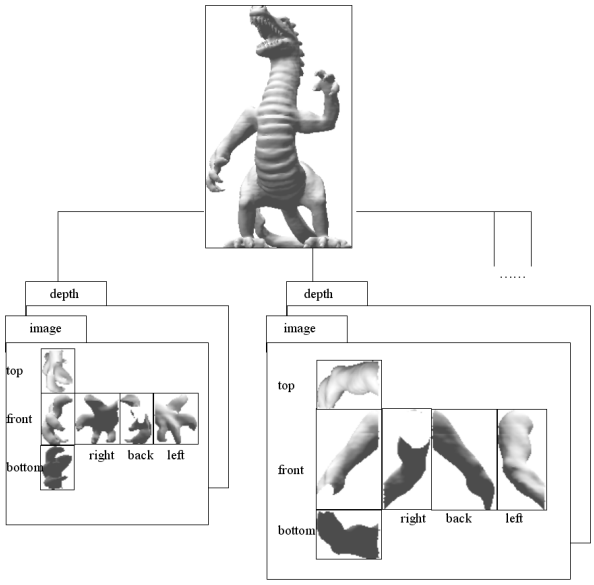

Fig. 5 The subdivision data structure.

3.2. Subdivision Criteria

The criteria for subdividing an object was as follows. In the spirit of addressing the visibility and hole problem, subdivisions were made for parts of the object with substantial concavity. Ideally, a convex decomposition can be applied for this purpose, however, we have done the subdivision manually. 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 subobject independence. For instance, if a motion behavior like rotating an arm, or moving a finger is expected of the object, that part would be designated as a subdivision (See Figure 7). A drawer of a desk, or a book in a bookshelf would be designated as a subdivision (assuming that the 3D model of the subobject exists). Finally, we have 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.

3.3. Rendering Algorithm

The rendering algorithm for the “subdivided” Relief Texture is exactly the same as the original one. The only difference is in that we repeat the 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). Figure 5 illustrates the concept. However, if the subdivision boxes overlap or not placed orthogonal to one another, it can be difficult to figure out the rendering order. We come back to issue in Section 3.6.

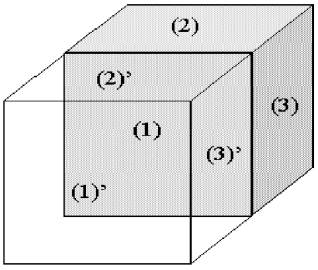

Fig. 6 The back-to-front ordering for rendering the subdivided Relief Textures. The back-to-front order is (1)-(2)-(3), then (1)'-(2)'-(3)'. The order among (1), (2), (3), or (1)', (2)', (3)' is unimportant.

3.4. Switching between Representations

Basically, the switching for subdivision should be based on the human visual or depth perception. That is, 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). In [4], we considered switching between the image and 3D models at a theoretical distance a human can start to detect depth within the object. However, maintaining a constant and acceptable (e.g. 10 ~15 Hz) frame rate is equally important in real time virtual environments. Given positions, P_A and P_O, of a user A and an object O, the angle between the view direction and direction toward the object, θ_{AO}, the relative importance of the object is formulated as follows.

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

$$(0 \le u, v, w, x \le 1)$$

$P(O)$ 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 are determined empirically.

3.5. Object Behavior

One of the goals of mixing (or maintaining) a dual representation was to allow more flexible implementation of object behavior. Although one straightforward way to do it is, when needed, 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 if the changed geometry happens to be contained in one subdivision, because if it is to be converted back to the image representations, it would have to be updated according to the result of the behavior. This is why careful subdivision would be needed at the modeling stage.

Sometimes, if the lighting effect can be ignored to some degree, we can simply apply 3D transformation or simple behavioral effect (e.g. change color) directly to the image representation (i.e. Relief Texture box) instead as shown in Figure 7. This can be most useful when a remote interactive behavior is needed (e.g. making an object selection by ray-casting, then activating some simple behavior).

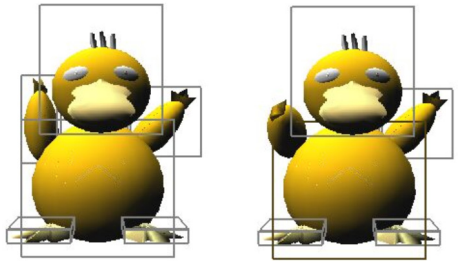

Fig. 7 Pokemon's right arm is rotated as a Relief Texture box.

3.6. Reducing the Overhead in Texture Processing

If the number of the object subdivision is too high, the overhead in the required texture operations can negatively affect the overall frame rate and defeat the purpose of subdivision in the first place. Figure 8 shows the change in the frame rates for two objects, one subdivided into 15 cells, and the other in 64 cells. The object is initially placed near the user (and exist as 3D polygonal model), moves away from the user (gradually transits into the Relief Texture representation), then comes back (converts back to the 3D representation). Even though the transition from one representation to the other is more or less smooth (no significant jumps), the graph (Figure 8(a)) shows a drop in the frame rate for the case of 64 subdivision object compared to that of the 15. To solve this problem, we merge all the textures of the children subdivisions, project them on the side of a virtual bounding box that includes all the subdivisions in consideration, and merge them as one texture (there would be three merged textures needed as only three textures are visible at one time). Figure 9 shows the process. This allows the subdivision bounding boxes to be overlapped or even placed in angles[1].

[1] If the subdivision bounding boxes are not orthogonal to one another, new depth values must be computed to correctly merge the textures.

Actually, this is similar to selecting the right subdivision level (in the subdivision tree) for a part of an the object, and representing it as one Relief Textured box. We can even avoid the merging operation by storing the texture images for intermediate nodes in the subdivision hierarchy. Then, the subdivision grouping cannot be chosen dynamically. Figure 8(b) shows that the frame rates for the two objects are now similar despite the difference in the number of subdivisions, after employing this process. Figure 10[2] illustrates the result (the dinosaur model has about 10,000 polygons).

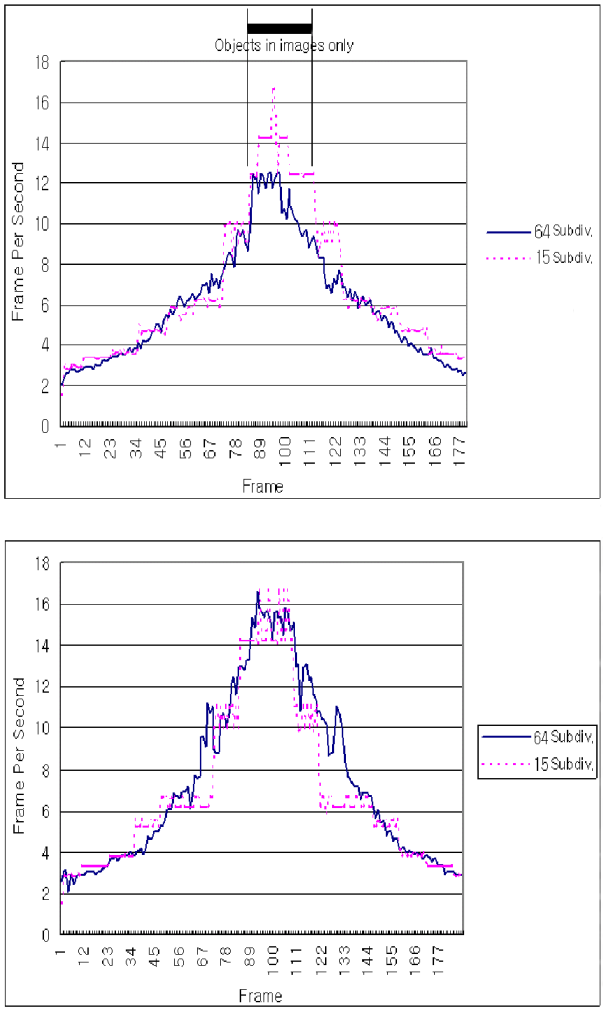

Fig. 8 (a) Frame rates for objects with 64 and 15 subdivisions (384 small textures vs. 90 large textures) when treating them separately. (b) Frame rates for the same two objects when textures are merged then warped. The object used is shown in Figure 5 which originally had about 10,000 polygons.

[2] All examples in the paper were run on a 700 MHz desktop Pentium III PC with the NVIDIA GeForce II graphics board.

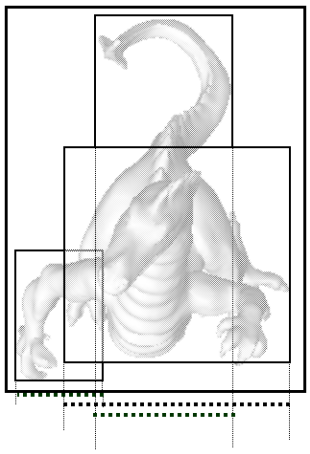

Fig. 9 Merging all subdivision textures on to a virtual bounding box to reduce texture operation overhead. The dinosaur object is seen from the top for illustration.

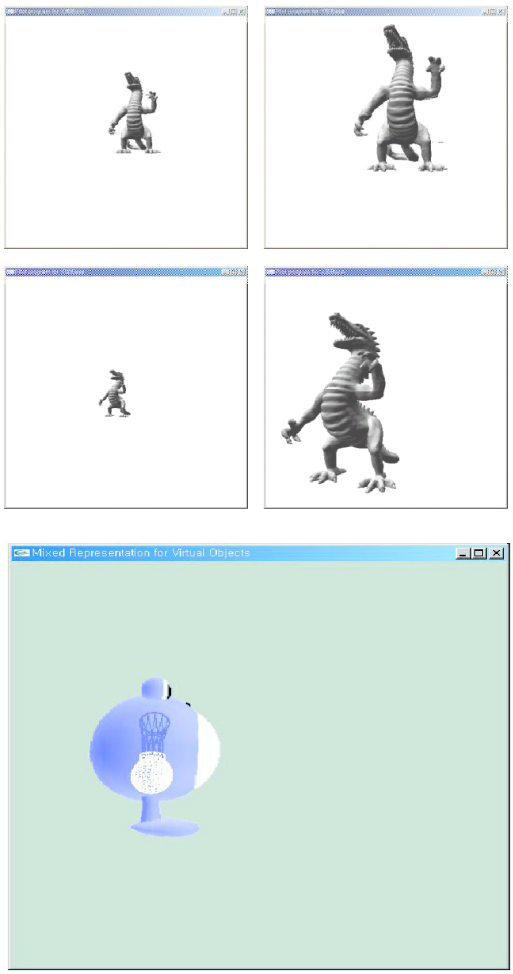

Fig. 10 A virtual object seen from four different view points. Depending on the viewing distance and other parameters, the object uses different mixes of representations, but no significant visual anomaly can be detected. The second example shows the same Pokemon of Figure 4 with its arm represented with polygons in a different view point.

4. Building a Virtual Environment

This section discusses two 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.

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. Figure 11 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. When the bounding boxes collide, we merge the two subdivision cells into a virtual bounding box in the same way explained in the Section 3.6 for reducing the number of texture operations.

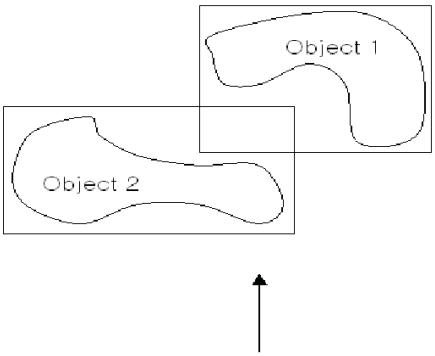

Fig. 11 Two different subdivision cells belonging to two different objects in collision. Object 1 occludes Object 2 in the middle when seen in the arrow direction.

Figure 12 shows a simple virtual environment with many virtual objects with the mixed representation (also See the accompanying video). 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, 5,000). Within this constraint, we apply the formula introduced in Section 3.4 at every frame to decide which objects (or cells) are important enough to be rendered in 3D polygons. Figure 13 shows the variation in the frame rate as a user moves around the scene. Note that the frame rate never drops below 10 fps (the variation above the minimum frame rate is much less problematic).

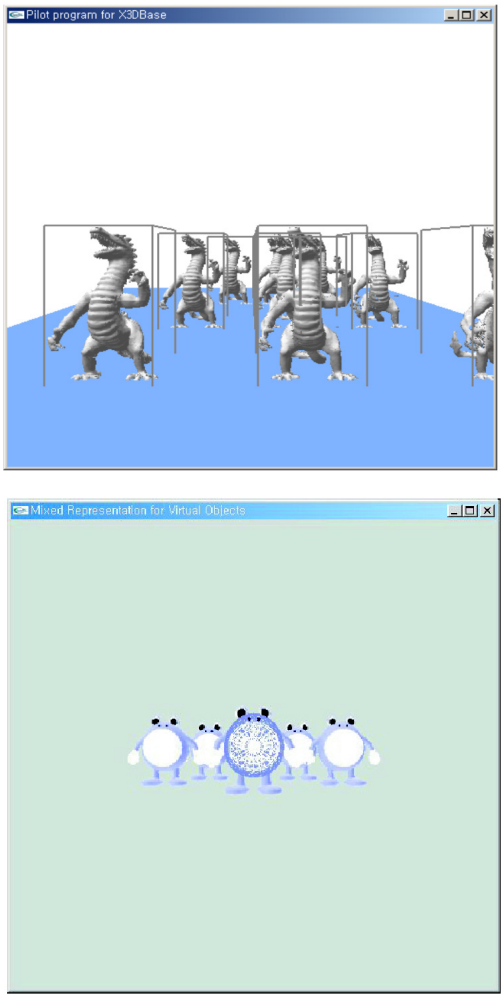

Fig. 12 A virtual environment with multiple virtual objects with mixed representation.

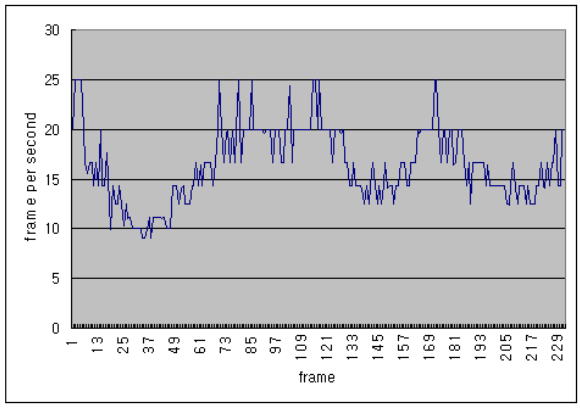

Fig. 13 The frame rate trend when a user navigates in the virtual environment of Figure 12. A dynamic switching of representations is applied to keep a target frame rate at 10 Hz.

5. Conclusion

In this paper, we presented an approach to mix both polygonal and image based representations in rendering a virtual object. The main motivation for doing this is in achieving real time rendering (with minimal frame rate variation) of virtual objects and allow some interactivity, especially when the objects are made of a high number of polygons relative to the polygon processing capability of a given graphics hardware. We further proposed to subdivide the object into cells according to its structure, polygon count, and visibility and use Relief Textures to implement the concept as efficient as possible. We demonstrated its potential in terms of its visual appearance and property that lends itself to an effective performance management for real time virtual environments. Our future work lies in improving the performance of the proposed method with a clever management of the potentially large image data counts. A more practical application of the proposed technique in an interactive virtual environment is in progress.

Acknowledgements

This project has been supported in part by the Korea Institute of Science and Technology (KIST), Ministry of Education's BK21 Project, and the Korea Science and Engineering Foundation supported Virtual Reality Research Center.

References

1. D. Aliaga and A. Lastra: "Automatic Image Placement to Provide a Guaranteed Frame Rate," *ACM SIGGRAPH 99 Conference Proceedings*, pp. 307-316, (1999).

2. D. Aliaga and A. Lastra: "Architectural Walkthroughs using Portal Textures," *Proc. of Virtual Reality Annual International Symposium*, pp. 228-233, (1998).

3. M. Rafferty, D. Aliaga, and A.Lastra: "3D image Warping in Architectural Walkthroughs," *Proc. of Virtual Reality Annual International Symposium*, pp. 228-233, (1998).

4. J. Yoon and G. Kim: "An Intergrated VR Platform with 2D Images and 3D Models," *Proc. of the Intl. Conf. on Virtual Reality and its Application in Industry*, pp 9-14, (2002).

5. G. Schaufler: "Dynamically Generated Imposters," *Proc. of the Workshop on Modeling Virtual World - Distributed Graphics*, pp. 129-136, (1995).

6. G. Schaufler: "Nailboards: A Rendering Primitive for Image Caching in Dynamic Scenes," *Proc. of the Eurographics Workshop on Rendering*, pp. 151-162, (1997).

7. M. Oliveira and G.Bishop: "Image-Based Objects," *Proc. of ACM Symposium on Interactive 3D Graphics*, pp. 191-198, (1999).

8. M. Oliveira and G.Bishop, David McAllister: "Relief Texture Mapping," *Proc. of ACM SIGGRAPH 2000 Conference*, pp. 359-368, (2000).

9. L. McMillan Jr: "An Image-Based Approach to Three-Dimensional Computer Graphics," *Ph.D Dissertation University of North Carolina at Chapel Hill (Also UNC Computer Science Technical Report TR01-019)*, (2001).

10. G. Tharp, A. Liu, L. French, S. Lai, and L. Stark: "Timing Considerations of Helmet-Mounted Display Performance," *Proc. of the SPIE - Human Vision, Visual Processing, and Digital Display III Vol. 1666*, pp. 507-576, (1992).

11. M. Reddy: "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*, (1997).

12. S. Chen and L. Williams: "View Interpolation for Image Synthesis," *Proc. of ACM SIGGRAPH 1993 Conference*, pp. 279-288, (1993).

13. S. Seitz and C. Dyer: "View Morphing," *Proc. of ACM SIGGRAPH 1996 Conference*, pp. 21-30, (1996).

14. S. Gortler et al: "The Lumigraph," *Proc. of ACM SIGGRAPH 1996 Conference*, pp. 43-54, (1996).

15. M. Levoy and P. Hanrahan: "Light Field Rendering," *Proc. of ACM SIGGRAPH 1996 Conference*, pp 31-42, (1996).

16. W. Dally et al: "The Delta Tree: An Object Centered Approach to Image based Rendering," *MIT AI Lab Technical Memo 1604*, pp. (1996).

17. J. Shade, S. Gortler, L. He, and R. Szeliski: "Layered Depth Images," *Proc. of ACM SIGGRAPH 1998 Conference*, pp. 231-242, (1998).

18. J. Neider et al: "OpenGL Programming Guide," *Addison Wesley Publishing Company*, (1993).