The Future of Mixed Reality: Issues in Illumination and Shadows

Charles E. Hughes, Jaakko Konttinen, Sumanta N. Pattanaik
School of Computer Science
University of Central Florida
Orlando, Florida
ceh@cs.ucf.edu, jaakko@cs.ucf.edu, sumant@cs.ucf.edu

ABSTRACT

Mixed Reality (MR) is a blending of real and virtual objects. How well that blending works is critical to a user's experience within an MR scenario. The focus of this paper is on the visual aspects of this blending; other senses such as sound and haptics are covered elsewhere.

Blending the real and virtual realities in MR requires that the virtual objects react properly to changes in real lighting and that the real react properly to the insertion of virtual lights (e.g., a virtual flashlight). Even more challenging, virtual objects must cast shadows on real objects and vice versa. Making this realistic means that all such interactions must occur at interactive rates (30+ frames per second).

Our research focuses on algorithmic development and implementation of these procedures on programmable graphics units (GPUs) found commonly on today's commodity graphics cards. The algorithms we develop are tailored to take advantage of the parallel pipeline architecture of GPUs and to carefully avoid some of the limitations found in currently available versions of these units.

ABOUT THE AUTHORS

Charles E. Hughes is Professor of Computer Science in the School of Computer Science at the University of Central Florida. After receiving his PhD in computer Science in 1970, he served on the faculties of Computer Science at Penn State University and the University of Tennessee prior to joining UCF in 1980. His research interests are in mixed reality, distributed interactive simulation and models of distributed computation. He has authored or co-authored over 100 refereed research papers and six books. He is a senior member of the IEEE, and a member of the ACM and the IEEE Computer Society.

Jaakko Konttinen is an undergraduate student in School of Computer Science at the University of Central Florida. He has served as a research assistant in the UCF/CS graphics group since Fall, 2003. His research interests are in using programmable graphics hardware to accelerate realistic rendering.

Sumanta N. Pattanaik received his Ph.D. degree in Computer Science in 1993 from BITS-Pilani, India. He was a researcher in IRISA/INRIA, France from 1993 to 1995 and in the Program of Computer Graphics at Cornell University from 1995 to 2001. Since July 2001, he has been an associate professor of Computer Science at University of Central Florida. He is a Category Editor (Computer Graphics) of ACM Computing Reviews. His current research interest is in real-time realistic rendering using programmable graphics hardware. He is a member of IEEE and ACM SIGGRAPH.

The Future of Mixed Reality: Issues in Illumination and Shadows

Charles E. Hughes, Jaakko Konttinen, Sumanta N. Pattanaik
School of Computer Science
University of Central Florida
Orlando, Florida
ceh@cs.ucf.edu, jaakko@cs.ucf.edu, sumant@cs.ucf.edu

INTRODUCTION

Mixed Reality (MR) covers the broad spectrum of mixing the real and virtual that runs from Augmented Reality (AR), where the virtual augments the real (e.g., where people and objects in the room may be a mixture of real and virtual), to Augmented Virtuality, where the real world augments the virtual (e.g., when real people appear to be situated in a virtual setting such as in a model of an urban environment). (Milgram and Kishino, 1997)

The blending of the visual aspects of the real world and virtual components is achieved in current MR systems by using one of two visual capture/display techniques. The first approach is to employ an optical see-through Head Mounted Display (HMD) with virtual objects inserted into the user's visual field (Rolland and Fuchs, 2000); the second is to employ a video see-through HMD in which the real world, as captured through cameras on the HMD, is processed, changed and augmented with virtual objects, and then transmitted to displays in the user's direct line-of-sight. (Uchiyama et al., 2002) Our work assumes the latter.

Employing Mixed Reality as the basis for commercial and educational products requires that complex virtual content be seamlessly merged with the real. (Stapleton et al., 2002) This blending requires an analysis and understanding of the real objects so that proper inter-occlusion, illumination, and inter-shadowing can occur. The issues addressed in this paper are: (a) Lighting of real by virtual and *vice versa*, and (b) Shadowing of virtual on real and *vice versa*. Audio and haptics, while equally important, are not addressed here.

MR/MOUT

Although the techniques we present here are applicable to all MR experiences in which lighting is important, illumination and shadows play a particularly critical role in training for military operations in urban terrain (MOUT). The research reported here is presently being integrated into the MR/MOUT project, a project supported by the U.S. Army's Science and Technology Objective (STO) Embedded Training for Dismounted Soldier (ETDS) at the Research, Development and Engineering Command (RDECOM). This, in turn, is being integrated with the Naval Research Laboratory's BARS System in a related project supported by the Office of Naval Research.

The primary issue in MR/MOUT is the recognition of potential threats by soldiers on the ground who are carrying our high-risk operations such as room clearing. Such threats are often heard (footsteps) and their shadows seen, long before direct visual contact occurs. To provide the realism required to properly train people in these MR environments, it is necessary that virtual characters (friendlies, neutrals and hostiles) cast shadows correctly in interactive time. This requires the correct rendering of all the combinations we have discussed, real on virtual and virtual on real, as well as the easy cases of real on real (nature does it) and virtual on virtual. Additionally, casting virtual light on real objects (e.g., with a virtual flashlight) and having real light effect the appearance and visibility of virtual objects provides the realism needed for successful training exercises within darkened buildings and in night settings.

APPROACHES

Accurate computation of illumination and shadow of virtual objects in Virtual Worlds is challenging because of issues of inter-object visibility and complex interaction of light with objects. However, the challenge in Mixed Reality is substantially greater. Here, we do not have control of all environmental conditions (*e.g.*, lighting) and we do not have any notion of the intent of the mobile real objects (*e.g.*, people). Depth from stereo and other depth cue techniques can help with the mutual occlusion problem, but do not provide any help in the proper illumination of virtual objects. Unfortunately, failing to consider illumination is one effect that makes virtual objects stand out from the real, appearing obviously synthetic. Additionally, differences in illumination (and positioning strategies) can have negative impacts such as haloing of the synthetic object(s).

In our research, we are developing efficient rendering algorithms that address both the effect of the real world on virtual objects and the effect of virtual objects on

the real world. To handle these issues we need at the very minimum real-time capture of the real-world illumination at every point of the virtual object, and real-time modeling of the real world. While we do not yet have a full solution to this problem, we have had substantial successes. We pre-design geometry of the visible real objects to simulate their shadow and inter-reflection effects on virtual objects and vice-versa. For a static physical world that is known in advance, this pre-designing process is acceptable. We capture real-world illumination as high dynamic range environment maps at a point of the scene using a camera specifically designed to capture the environment. (Pointgrey, 2004) If we assume that the major light source direction does not change significantly, then this captured illumination can be used for lighting all the virtual objects in the environment.

LIGHTING AND SHADOWS IN MR

Our proposed method for lighting and shadow in MR environments is based on conventional, strictly VR lighting techniques that have been adapted to work with real objects in a MR environment. We require two things to be known of the real objects at the time that lighting is calculated: geometric information of the real scene, and camera pose information.

PHANTOM MODELS

We pre-model and represent geometric information of real objects in the scene by "phantom" models, which are often used as occlusion models. These are simple triangle meshes that are never really drawn on screen, but still contain information about the real object's surface, such as vertex positions, tangent directions and normal directions, that are required for occlusion and lighting calculations. A later section of this paper describes an interactive method to create phantoms for planar objects such as table tops, floors and walls.

When used as occlusion models, invisible renderings of phantom objects visually occlude other models that are behind them, providing a simple way to create a multilayered scene, e.g., with the model of a person inside a building only visible through portals (doorways and windows). When used for lighting and shadows on real objects, these models help us calculate shading changes for their associated pixels. Thus, using them, we can increase or decrease the effects of lights, whether real or virtual on each affected pixel. Decreasing simulates shadows from interfering objects; increasing simulates directional lighting. Alternatively, we can decrease lighting and then add it back in as necessary.

CAMERA TRACKING

In addition to geometry, we need the spatial relationship between the virtual lights, the real objects, and the camera. This is required for most lighting calculation and is needed so we can superimpose phantom objects on the corresponding real objects. For this to be possible, we must be able to track the position and orientation of the camera in the coordinate space of the phantom objects.

Tracking can be done by adding tracking probes to important objects or by analyzing a scene, usually based on shape recognition. Tracking probes can involve magnetic, acoustical or optical detection (active LEDs or passive markers). Our approach is not tied to any specific tracking technique so long as it provides the required alignment transformation. In this paper, we show examples that use tracking based on shape marker detection (Figure 1).

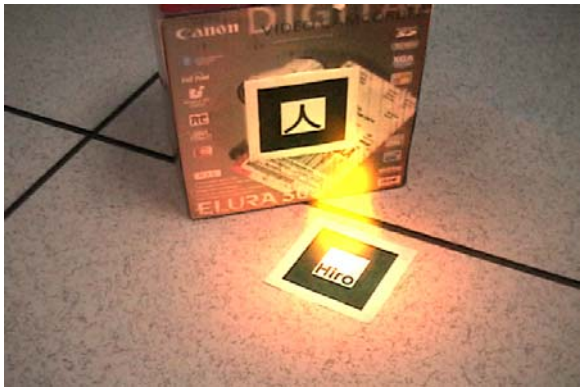

Figure 1: Virtual fire illuminating a real world.

ILLUMINATION

We perform the actual illumination by shading the original pixel color from the image based on the lighting calculation. Because of this, we are restricted to illuminating only those pixels for which we have geometric information in the form of phantom objects transformed into image space. The one exception to this is when we want to change only the amount of ambient virtual light in the scene.

As calculating lighting contributions is computationally intensive, we do these calculations on programmable fragment shaders found in modern commodity graphics hardware such as those manufactured by ATI and nVidia. These graphics processing units (GPUs) are, in effect, small parallel computers, providing both SIMD and pipeline parallelism.

We perform the actual shading operation by using alpha blending between the lighting contribution and

the original pixel intensity. The blending parameters depend on the effect we want to accomplish. Suppose that vector D defines the original pixel color with three components for the RGB channels and one for alpha. Similarly the vector *S* defines the virtual lighting contribution for that pixel from the fragment shader. We can then define the final color *C* as:

$$C = D + M * S$$

where *M* represents the material reflectance properties of the surface at that point. Since we rarely have access to material property information for real objects, we can approximate this property by using the original pixel color in place of *M*. The desired equation is then:

$$C = D + D * S \quad (1)$$

If *S* has a range of [0, 1], then the above equation is equivalent to scaling *D* by a factor between [1, 2]. Expressing this in the form of eq. (1) is well-suited for use with alpha blending operations.

To decrease an image's brightness, we simply scale each channel down by some constant factor. Restoring intensity then becomes a matter of modulating the darkened pixel color by the light contribution from the virtual light. The alpha blending parameters for this are the same as for the above operation.

Figure 1 uses virtual fire to illuminate a real environment based on the brightening method described in the previous paragraphs. The motivation behind this example was to see if a highly dynamic light source such as fire could still convincingly illuminate a real environment.

For the lighting calculation, we chose a point light-based approximation of the light contributed by all particles. We sorted each particle into separate groups based on the particle's remaining life, and used the average positions and intensities in each group to calculate a point light for that group. The total light contribution is then the sum of the light contribution from each point light. We brighten surrounding pixels based on these point lights. A more physically accurate lighting model would certainly give much better results and could be implemented without having to change the underlying shading method.

INCLUDING SHADOWS

In this section we describe a method for adding shadow to a scene lit by virtual lights. The method for adding shadow is based on (Haller, 2003) and has been modified to include light contributions from virtual lights. The method is based on the stencil shadow volume technique (Crow, 1977).

Shadow volumes

Given a point light source and an occluding object, a shadow volume defines the portion of space that is in the occluder's shadow. Any point that lies inside this volume is not lit by the given light source.

To construct a shadow volume for some combination of light and occluder, we begin by finding the silhouette set of edges for the occluder. The silhouette edge set is the set of those edges that would appear in the silhouette of the shadow. One simple method of finding silhouette edges for triangle meshes is to iterate through each edge of the occluder while looking at the facings of the two triangles between which the edge lies. If one triangle faces the light and the other faces away from the light, then the edge is a silhouette edge.

This silhouette edge set creates the "outline" of the shadow volume. We now need to extrude this shape into a three-dimensional volume. This is performed by considering each edge and performing the following operations. Each edge is defined by two vertices. For each of these vertices, we construct a vector from the light to the vertex, and duplicate that vertex along the vector some distance away from the light. The distance can be arbitrarily determined, and most implementations use the light's maximum range as the distance. Now we have another version of each silhouette edge some distance away from the light. Each pair of edges, the extruded and the unextruded edge, forms two sides of a rectangle. The remaining two sides can be easily constructed by forming two new edges from each corresponding pair of edge vertices. If we construct such a rectangle for each pair of edges, we have created a solid volume.

Testing if a point lies within an arbitrary volume can be an expensive operation. For improved performance, we carry out this calculation in graphics hardware.

Stencil shadow rendering

The stencil shadow volume rendering technique is a hardware-accelerated approach to testing if a pixel lies inside some shadow volume or not. It is well-supported by most video cards because the only special feature required is stencil buffer support. The stencil buffer is an extension of the depth buffer and is used to stencil out certain areas from rendering. When writing to the stencil buffer, arithmetic operations can be performed on the data values. The frame buffer can

then be initialized to only render on those parts that pass a user-definable comparison test with the matching stencil buffer data value. In a strictly virtual setting, the technique consists of the following rendering passes:

1. Render all objects with ambient lighting only.
2. Fill the stencil buffer based on the calculated shadow volumes.
3. Render those parts with full lighting that are not in shadow.

Stages two and three are not very intuitive and require some clarification. First note that in the process of rendering all objects with ambient lighting in step one, we have filled the depth buffer with the final depth information for the scene. We thus disable writing to the depth buffer for the remaining passes, although it is crucial to the technique that we still perform depth testing.

Step two is performed by first clearing the stencil buffer to all zeroes, then rendering each shadow volume to the stencil buffer in two parts. In the first pass, we only render front-facing polygons, and for all visible shadow volume pixels we increment the stencil value by one. In the second pass, we render only back-facing polygons and decrement the stencil value by one. After all shadow volumes have been rendered, the pixel is in shadow if the stencil value is non-zero.

This works because of the depth information from step one. It is effectively the same as casting rays from the eye through each pixel on the image plane, and terminating the ray on the first shadow-receiving object. If the pixel is in shadow, the point of termination of the ray must lie within the shadow volume. Another way of looking at this is that the ray entered the shadow volume but never exited it. So for all pixels where the shadow volume intersects a shadow-receiving object, front-facing triangles pass the depth test but back-facing triangles fail it, and the addition to the stencil value is never negated.

Adapted version

The standard stencil shadowing technique (Haller, 2003) only deals with virtual-to-virtual shadowing, but we must also consider shadows cast from virtual to real objects and vice versa. The effects of including shadowing with illumination give most appealing results when combined with the image pre-darkening method from earlier section. Changes required for implementing shadows in a situation with no ambient darkening are mentioned where necessary. The steps for rendering shadows are as follows:

1. Render Real shadows on Real objects.
2. Render Virtual shadows on Real objects.
3. Render Real and Virtual shadows on Virtual objects.

In this notation, real objects denote the phantom objects of each tracked real object. Each step is discussed in more detail in the following sections. Real shadows on real objects are already contained in the captured image, so step 1 can be executed by simply drawing the captured image. The remaining steps are discussed in the following sections.

Virtual/Real shadows

The purpose of this step is to render shadows cast from virtual objects to real objects. In the process we will also virtually darken the image, and then restore intensity to real objects with the virtual light. It is basically an execution of the standard stencil shadow volume algorithm with some additional steps:

1. Render real object phantoms to depth buffer.
2. Darken areas not covered by phantoms by factor *F*.
3. Render shadow volumes to stencil buffer.
4. Light areas that are not in shadow by the virtual light according to above sections.
5. Darken areas in shadow by factor *F*.

To determine the set of shadow volumes to render in stage three depends on whether or not we want real objects to cast new shadows on other real objects. If we do not, then the total set is only the set of virtual shadow volumes. If we do, then we render real shadow volumes as well.

In stage four, we use one of the blending operations from one of the previous sections. If we are restoring intensity, the thing to remember is that the destination pixels are still at their maximum intensity. If we are modulating the destination color by the incoming color (which is the lighting contribution), then we should choose an ambient color that matches *F*. If we are increasing intensity, then stages two and five can be ignored.

Figure 2 shows a demonstration of a virtual light illuminating virtual and real objects, and virtual objects

casting shadow on virtual and real objects. We track a special marker which represents the location of the virtual flashlight in the scene. The user can "shine" the flashlight at real objects which should then be lit correctly. The flashlight should also illuminate any virtual objects it is shined towards.

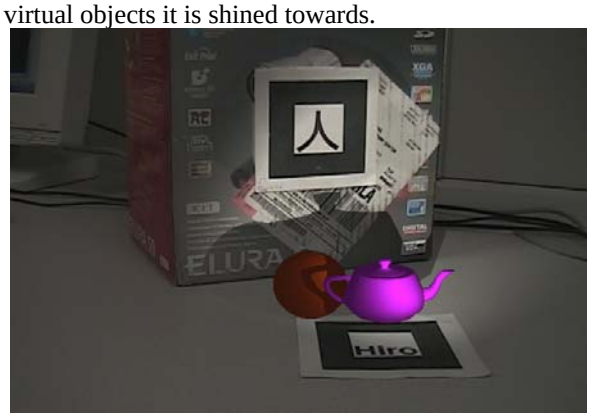

Figure 2: Virtual flash light illuminating virtual and real objects.

INTERACTIVE PHANTOM GENERATION

The main motivation of using the following method is easy adaptability to a new testing environment, which means that the phantom geometry for real objects has to be easily recalculated on the testing site. For simplicity we restrict ourselves to constructing planar surfaces that were on the plane of the marker.

Suppose the transformation is represented as a 4x4 matrix **M**, then the equation for screen-space coordinates x and y from world space coordinates X, Y and Z where Z=0, are represented by the following equations:

$$x = \frac{M_{11}X + M_{12}Y + M_{14}}{M_{41}X + M_{42}Y + M_{44}}$$

$$y = \frac{M_{21}X + M_{22}Y + M_{24}}{M_{41}X + M_{42}Y + M_{44}}$$

Now, for any given x and y, we can solve for the corresponding X and Y by solving the following system of linear equations:

$$A = BX + CY$$
$$D = EX + FY$$

Where

$$A = M_{44}x - M_{14} \quad D = M_{44}y - M_{24}$$
$$B = M_{11} - M_{41}x \quad E = M_{21} - M_{41}y$$
$$C = M_{12} - M_{42}x \quad F = M_{22} - M_{42}y$$

Our software allows the user to quickly trace a concave polygon in the image where each vertex is defined in screen space, use the above algorithm to solve for the shape of the polygon in world space, and then convert it to a triangle mesh which can be lit through the use of a triangulation algorithm. We assume that the vertex normals are always perpendicular to the polygon. Because the tracing happens entirely in screen space, it could be fully automated using some feature tracking algorithm. Unfortunately our method is limited to planar shapes. Calculating the changing Z-coordinate of a non-planar shape would require employing dense stereo data or some Computer Vision-based technique such as structure from motion.

ENVIRONMENT LIGHTING

Unlike point light sources, illumination from a scene is omni directional in nature and hence rendering of virtual objects with such a light source is not straightforward. We pre-compute and store accurate lighting effects of spherical harmonics basis light sources on the vertices of the virtual objects. At the time of rendering for MR, we approximate the captured environment light into a linear combination of the spherical harmonics basis. We make use of the GPU vertex engine to compute the lighting at each vertex by modulating the stored lighting effects corresponding to each spherical harmonics basis light with the corresponding approximation coefficient and summing the modulated values.

The images in Figure 3 show a virtual object (bunny) accurately lit using the captured light of the scene. For shadow computation on the table, we well-tessellate the phantom geometry attached to the bottom of the virtual object. For each vertex of the phantom mesh, we pre-compute the lighting effect of the spherical harmonics basis lights with and without the virtual object. We store the ratio of these coefficients at the mesh vertices. During rendering, we carry out the same computation at the phantom mesh vertices as we do at the vertices of the virtual object. However, the computation result at the mesh vertices gives the attenuation factor. We attenuate the intensity of the pixels corresponding to the phantom by the interpolated factor. This results in a smooth shadow appropriate to the lighting in the scene. We demonstrate this shadow in Figure 3, left side image.

Notice the realistic shadow appearance on the table around the bunny in the left image. The image on the right is without shadow.

Figure 3: Accurately illuminated virtual bunny inserted into the real scene as viewed from a video see through HMD. The bunny on the left casts a shadow on the tabletop.

ACKNOWLEDGEMENT

This work is partially supported by the Office of Naval Research, ATI Research, the I-4 Corridor Fund and the Army's Research, Development & Engineering Command (RDECOM, Orlando). Special thanks are due to the Mixed Reality Laboratory, Canon Inc., for their generous support and technical assistance.

REFERENCES

1. Crow, F. C. (1977). Shadow Algorithms for Computer Graphics, *Proceedings of SIGGRAPH 77*, 242-248.
2. Haller, M., Drab, S., & Hartmann, W. (2003). A Real-time Shadow Approach for an Augmented Reality Application Using Shadow Volumes, *Proceedings of ACM Symposium on Virtual Reality Software and Technology (VRST 2003)*, 56-65.
3. Milgram P. & Kishino, F. (1994). A Taxonomy of Mixed Reality Visual Displays, *IEICE Trans. on Information and Systems, E77-D(12)*, 1321-1329.
4. Pointgrey Research Inc. (2004). Retrieved June 20, 2004, from http://ptgrey.com/products/ladybug/.
5. Rolland, J.P., & Fuchs, H. (2000). Optical Versus Video See-Through Head-Mounted Displays in Medical Visualization," *Presence: Teleoperators and Virtual Environments 9(3)*, 287-309.
6. Stapleton, C. B., Hughes, C.E., Moshell, J. M., Micikevicius, P., & Altman, M. (2002). Applying Mixed Reality to Entertainment, *IEEE Computer 35(12)*, 122-124.
7. Uchiyama, S. et al. (2002). MR Platform: A Basic Body on Which Mixed Reality Applications are Built, *ISMAR 2002*, Darmstadt, Germany, 246-256.