The Future of Mixed Reality: Issues in Illumination and Shadows

Jaakko Konttinen
School of Computer Science
University of Central Florida
Orlando, Florida
jaakko@cs.ucf.edu

Charles E. Hughes
School of Computer Science
University of Central Florida
Orlando, Florida
ceh@cs.ucf.edu

Sumanta N. Pattanaik
School of Computer Science
University of Central Florida
Orlando, Florida
sumant@cs.ucf.edu

Military training, concept design and pre-acquisition studies are often carried out in virtual settings in which one can experience that which is, in the real world, too dangerous, too costly or even beyond current technology. Purely virtual environments, however, have limitations in that they remove the participant from the physical world with its visual, auditory and tactile complexities. In contrast, Mixed Reality (MR) seeks to blend the real and synthetic. How well that blending works is critical to the effectiveness of a user's experience within an MR scenario. The focus of this paper is on the visual aspects of this blending or, more specifically, on the interactions between the real and virtual in the contexts of proper inter-occlusion, illumination, and inter-shadowing. This means that the virtual objects must react properly to changes in real lighting and that the real must react properly to the insertion of virtual lights (e.g., a virtual flashlight or a simulated change in the time of day). Even more challenging, virtual objects must cast shadows on real objects and vice versa. The proper casting of shadows is critical to military training, in that shadows often provide clues of others' movements, and of our own to others, long before visual contact is made. Realistic shadows can improve training greatly; their omission or the insertion of physically incorrect shadowing can lead to negative training. To be effective, visual realism requires that all such interactions 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. Our primary application is training of dismounted infantry for the complexities of military operations in urban terrain (MOUT).

Keywords: mixed reality, virtual reality, real-time rendering, illumination, graphics processing unit (GPU)

1. Introduction

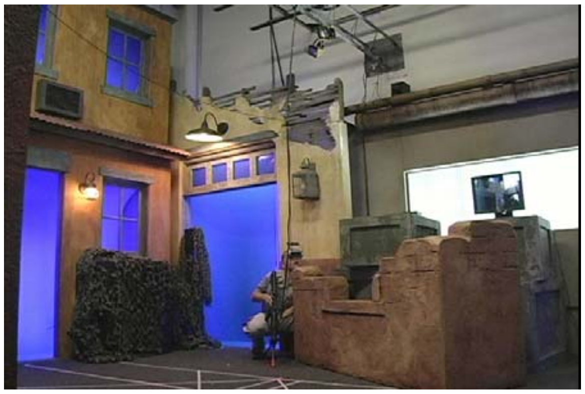
Mixed Reality (MR) covers the broad spectrum of mixing the real and virtual that runs from Augmented Reality (AR), where the virtual augments the real (e.g., where people and objects in the room may be a mixture of real and virtual), to Augmented Virtuality, where the real world augments the virtual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

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 [8]; 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 [10]. Our work assumes the latter.

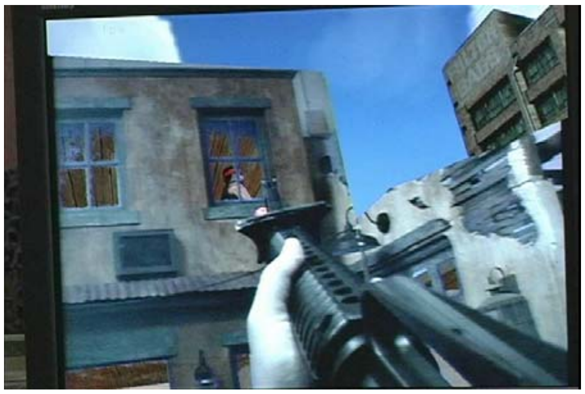
Employing Mixed Reality as the basis for commercial and educational products requires that complex virtual content be seamlessly merged with the real [9]. This blending requires an analysis and understanding of the real objects so that proper inter-occlusion, illumination, and inter-shadowing can occur. 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 to the effectiveness of MR experiences, are not addressed here.

2. 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 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. See Figure 1.



a) Physical reality



(b) Augmented reality

Figure 1. MR MOUT

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.

3. 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 [6]. 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.

4. 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.

4.1. 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 plain triangle meshes that are never drawn on screen, but are used to derive information required for the occlusion of virtual objects by real objects, and the shading of real objects by virtual lights.

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). The renderings are invisible since the visual image of each phantom model’s real world counterpart is already contained in the captured video frame. When used for lighting and shadows on real objects, these models give us 3D information about the real world surface at each pixel they cover, which helps us calculate shading changes for those 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.

4.2. 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 calculations and is also needed for superimposition of phantom objects on their corresponding real objects in the image for the purpose of correct occlusion. For this to be possible, we must be able to track the 3D position and orientation of the camera in the coordinate space of the phantom objects (or vice versa).

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 [4] (Figure 2).

4.3. 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.

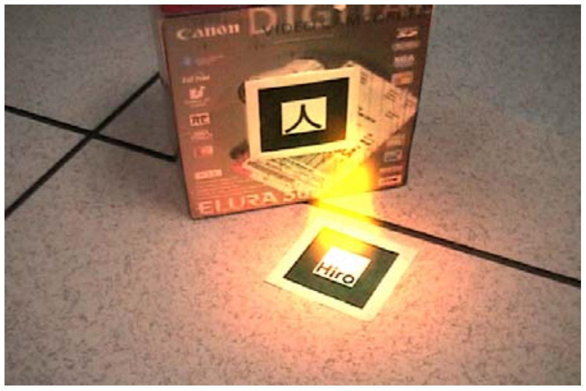


Figure 2. Virtual fire illuminating a real world

As calculating lighting contributions can be computationally intensive for complex surface materials or lighting distributions, 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.

Composition of contribution from the virtual lights into the video frame is done 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 (color, BRDF[1], etc.) of the surface at that point. Since we rarely have access to accurate material property information for real objects in a highly dynamic scene, we can approximate these properties by using the original pixel color in place of M. The desired equation is then:

$$C = D + D * S \tag{1}$$

The corresponding alpha blending parameters are shown below in the form of an Effect file commonly used to describe shading operations:

```
pass VirtualFire
{
      AlphaBlendEnable = true;
      SrcBlend = destcolor;
      DestBlend = one;
      ...
}
```

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. We have found this approximation gives adequate results for matte objects as shown in the figures. For specular objects, some estimation of the object's BRDF should be provided to maintain consistency with the real-world highlights and highlights from virtual lights.

To decrease an image's brightness, we scale each channel down by some constant factor.

[1] BRDF is the "Bidirectional Reflectance Distribution Function." It gives the reflectance of an object as a function of illumination and viewing geometry.

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.

For better results, the Gamma correction of the camera should first be inverted before the RGB values of the video frame are manipulated. After manipulation, Gamma correction should be re-applied to prepare it for display. This assumes that the camera follows the sRGB curve.

As an example, Figure 2 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 framework.

4.4. INCLUDING SHADOWS

In this section we describe a method for adding shadow to a scene lit by virtual lights. The method for adding shadow to an Augmented Reality scene is based on a novel method presented in [2] and has been modified to include light contributions from virtual lights. Haller's original method uses the shadow volume technique [1] from computer graphics adapted to hardware-accelerated graphics in [3].

4.4.1. Shadow volumes

Given a point light source and an occluding object, a shadow volume defines the subset of 3D space that is in the occluder's shadow. Any point that lies inside this volume is not lit by the given light source. Only information about the boundary of the shadow volume is necessary for our algorithm.

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 occluder. One 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 shared by the edge. 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 are 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.

4.4.2. Stencil shadow rendering

The stencil shadow volume rendering technique [3] 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.
```

Stage one lights the scene with ambient lighting. Ambient light is represented as a constant term and is used as a cheap approximation of indirect light reflecting from other objects, and is assumed to be unoccluded. 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 for the current frame. 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. Depth testing is a general hidden surface removal algorithm. For every pixel in the image, the depth buffer stores the distance of the closest point whose projection onto the image plane lies on that pixel. Thus when a new point is projected, it is *tested* against the current value in the depth buffer and discarded if the current value in the buffer is lower (i.e. a previously drawn point appears in front of the new one along the viewing ray through that pixel). Otherwise, the pixel is drawn on-screen and the depth buffer value is updated.

Step two is performed by first clearing the stencil buffer to all zeroes, then rendering each shadow volume boundary mesh to the stencil buffer in two parts. In the first pass, we only render front-facing polygons, and for all visible shadow volume pixels after depth testing 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 from the visible front faces to the stencil value is never negated by the subtraction from the culled back faces. The algorithm is similar to the point-in-polygon algorithm from computational geometry, where a point is determined to be inside a polygon by counting the number of times a ray to the point crosses the polygon boundary.

4.4.3. Adapted version

The standard stencil shadowing technique for Augmented Reality [2] only deals with shadowing in an environment where virtual lights are not expected to interact with real objects for purposes other than casting shadows. 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 (such as the case of the virtual fire example). We will adopt the terminology from [2] for describing the algorithm. The steps for rendering shadows with the adapted method 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.

4.4.4. 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.
```

Stage one fills the depth information for the scene.

Stage two begins with the assumption that everything in the scene for which we do not have 3D information is in shadow. If we are darkening the scene, this effect is performed here for those pixels that are not covered by phantom objects.

In stage three we determine which pixels for which we have depth information are in shadow. 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 when influenced by virtual lights. 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 for which we had depth information are still at their original (maximum) intensity. If we are modulating the destination color by the incoming color (which represents the lighting contribution), then we should choose an ambient color that matches *F*, the scaling factor from stage two. If we are increasing intensity, then stages two and five can be ignored.

In stage five, we perform the same operation as in stage two for those pixels that were determined to be in shadow by stage three.

4.4.5 Real/Virtual Shadows

The pass for rendering shadows from real and virtual objects to virtual objects remains unchanged. Briefly:

```
1. Clear the stencil buffer.
2. Render virtual objects with
   ambient lighting only.
3. Render real and virtual shadow
   volumes to stencil buffer.
4. Render unshadowed portions of
   virtual objects with full
   lighting.
```

Stages two and four should pick the same scaling factor *F* from 4.4.4.

Figure 3 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.

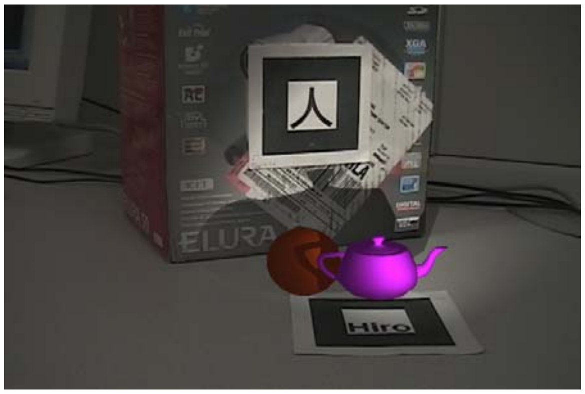


Figure 3. Virtual flash light illuminating virtual and real objects.

4.5. 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 4 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 4, left side image.

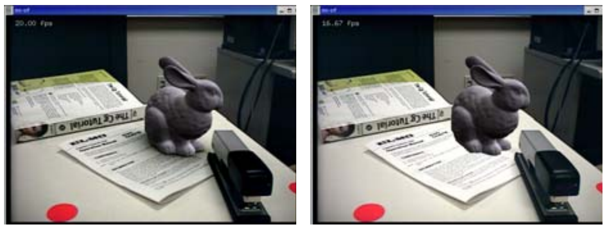


Figure 4. Accurately illuminated virtual bunny

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

4.6. SHADOW MAPPING

The advantage of the shadow volume method from 4.4 is that it always guarantees artifact-free shadows. Most mixed reality scenes however are highly dynamic, which means that the shadow volumes will most likely be recomputed every frame for many objects. If the objects are high polygon, this may affect performance.

An alternative shadowing method from computer graphics is shadow mapping. In shadow mapping, the view is rendered from the perspective of each light to an off-screen buffer called a shadow map and the output is distance to light instead of color. In the end, the shadow map's contents represent the distance of the first intersection point with an occluder for a particular ray of light. When rendering the scene from the camera's perspective, the shadow map is bound as a texture and for each visible point the matching pixel to which that point projects on the shadow map is found. If the distance sampled from the shadow map is less than the distance to the light of that surface point, the surface point is in shadow. This is analogous to a depth test from the light's perspective.

Shadow maps are useful since they can be used with any geometry that can be rasterized on screen, and they do not require expensive pre-computation on the CPU, such as shadow volume calculation. The disadvantage is that the resolution of the shadow map is finite and thus blocky shadow edges will be visible when viewed up-close.

Adaptation of the shadow mapping algorithm to mixed reality applications is simple because we already have geometry for real objects in the form of the phantom meshes. In short:

```
1. Render real and virtual objects
   to the shadow map.
2. Optionally darken all pixels
   with no depth information from
   phantoms by some scaling factor
   F.
3. Render real object phantoms on
   screen to fill the depth buffer.
   In this stage, change the pixel
   color based on virtual lighting
   contributions and the shadowing
   term from the shadow map
   algorithm.
```

```
4. Composite, light and shadow
   virtual objects on-screen.
```

In stage one, the shadow map is constructed as normally by rendering all virtual objects and all real objects from the light's perspective. All real objects are then rendered on screen with full lighting and modulated by the shadowing term given by the shadow map algorithm in stages two and three. All virtual objects are then composited to the scene and lit and shadowed similarly.

5. User-assisted Phantom Generation

The main motivation for using the following method is easy adaptability to a new testing environment, which means that the phantom geometry for real objects can 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

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

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 formula to solve for the shape of the polygon in world space, and then convert it to a triangle mesh that can be lit through the use of a general concave polygon triangulation algorithm. We assume that the vertex normals are always perpendicular to the polygon. Because the tracing happens entirely in screen space, it can 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 a computer vision-based algorithm such as structure from motion.

6. Delivering the Training Experience

The lighting and shadowing algorithms just described have been incorporated into a suite of software, the MR Software Suite (MRSS), which acts as our development and delivery system for MR experiences [6]. It integrates a collection of concurrent cooperating components. The central component is the MR Story Engine, a container for agents (actors), one for every user, virtual object and real object that interacts with other agents, plus additional agents that are useful for the story line. The other three subsystems (Figure 5) are for various aspects of rendering a multimodal simulation (Graphics, Audio and Special Effects).

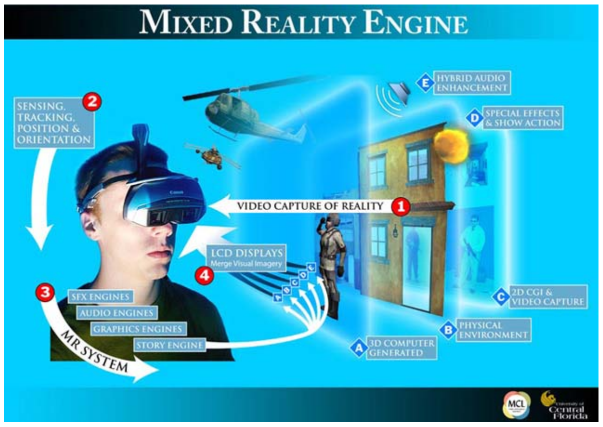


Figure 5. Flow of major MRSS components

The Graphics Engine is the part that contains the implementations of the lighting and shadowing algorithms. These visual effects, along with complex, realistic behaviors, significantly enhance the effectiveness of MR-based military training for encounters in close quarters.

The MRSS also provides a capture capability that is essential for "after-action review," a process used in training to assess the performance of users. Replaying trainees' actions can show them places where they missed cues (e.g., the shadows of hostiles) or where they provided cues that could or did place them in jeopardy. Moreover, the ability of our replay to change a user's viewpoint can be used to show trainees advantages that they may have gained if they had taken advantage of their environments, e.g., by standing in shadows

7. Conclusions and Future Directions

We have presented a method for including contributions from virtual lights in a mixed reality scene in a manner where computer graphics lighting algorithms can easily be integrated into a mixed reality application with few to no modifications to the existing mixed reality framework. We will now outline some potential directions of research to further enhance the effects of virtual lighting in the simulation experience.

The algorithms presented here assume information about real world surfaces in the form of pre-computed geometry. For highly uncontrolled scenes, such as those in which objects frequently undergo non-rigid transformations (e.g. bending), this information may be inadequate. A solution presents itself in the form of dense stereo data, which approximates the scene as a dense cloud of 3D points. The problem can then potentially be viewed in the domain of point-based rendering, where real-time rendering algorithms have recently appeared.

For real world objects with complicated material properties, more information is needed for believable virtual lighting to occur, such as measured BRDF data. Equipment and algorithms are available for extracting and compressing such information. If some limited knowledge of the real world lighting distribution is available, some on-the-fly estimation of the reflective properties of objects may also be performed with the use of high-dynamic range cameras.

The effective integration of lighting and shadowing is required to fully immerse a dismounted soldier into an MR training experience. However, this integration needs to go beyond the visual rendering described earlier. In particular, the behaviors of virtual objects need to be affected by lighting, just as we hope the behaviors of the human trainees are. That means, for instance, that, when the graphical rendering of a virtual entity casts a shadow, or the shadow cast by another virtual or real entity is in its line-of-sight, the agent associated with that entity must be "aware" of these circumstances. How the agent reacts is dependent on its behavior scripts. For instance, the agent may hide from a perceived threat, or it may ignore this state information if it is "dumb" or if other state information takes precedence, e.g., when it is part of an active fire fight.

To date, we have implemented some primitive feedback of visual rendering on agent behavior, e.g., using ray tracing to inform agents of objects in their line-of-sight. Creating more cognizant agents that react appropriately to feedback associated with lighting and shadowing is an active area of our current research and one which we believe will greatly increase the effectiveness of MR training.

8. Acknowledgements

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.

9. 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 VRST 2003*, 56-65.

[3] Heidmann, T. 1991. Real Shadows, Real Time. *Iris Universe*, 18 (November), 23-31.

[4] Kato, H. et al. 2000. Virtual Object Manipulation on a Table-top AR Environment. *Proceedings of ISAR 2000*, Munich, Germany, 111-119.

[5] Milgram P. & Kishino, F. (1994). A Taxonomy of

Mixed Reality Visual Displays. *IEICE Trans. on Information and Systems*, E77-D(12), 1321-1329.

[6] M. O'Connor, and C. E. Hughes, "Authoring and Delivering Mixed Reality Experiences," *Proceedings of 2005 International Conference on Human-Computer Interface Advances in Modeling and Simulation (SIMCHI'05)*, New Orleans, January 23-27, 2005, pp. 33-39.

[7] Pointgrey Research Inc. 2004. Retrieved June 20, 2004, from http://ptgrey.com/products/ladybug/.

[8] 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.

[9] Stapleton, et al. 2002. Applying Mixed Reality to Entertainment. *IEEE Computer* 35(12), 122-124.

[10] Uchiyama, S. et al. 2002. MR Platform: A Basic Body on Which Mixed Reality Applications are Built. *ISMAR 2002*, Darmstadt, Germany, 246-256.