Distributed Ray Tracing of Mirages

Michael D. Wilson*
University of Maryland, Baltimore County

Abstract

To further increase the realism of ray tracing, we take the idea of distributed ray tracing and add on to it mirage rendering. Mirages are seen quite often in our daily lives, and in order to make computer generated images look as realistic as possible, it is necessary to add things, such as mirage rendering, that are seen in the natural world.

Mirages are caused by light passing from colder air to warmer air. When this happens, the light bends in relation to the temperature gradient [Wikipedia 2007]. Mirages can be most often found on roadways and on the hoods of cars, looking somewhat like a liquid. In this paper, mirage rendering will be implemented using a technique described by Berger, Trout, and Levit wherein the parabolic equation that mirages adhere to is approximated using a multilayered box capable of internal reflectance and refraction [1990].

CR Categories: I.3.3 [Computer Graphics]: Picture/Image Generation; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism;

Keywords: ray tracing, refraction, mirage

1 Introduction

To introduce ray tracing, one should first look at the intensity equation related in Cook's paper [1984]:

$$I(\phi_r, \theta_r) = \int_{\phi_i} \int_{\theta_i} L(\phi_i, \theta_i) R(\phi_i, \theta_i, \phi_r, \theta_r) d\phi_i d\theta_i$$

where

L	=	the illumination function
R	=	the reflection function
(ϕ_i, θ_i)	=	angle of incidence
(ϕ_r, θ_r)	=	angle of reflection

Most ray tracing attempts today approximate this equation via a variety of methods, typically using simplifying assumptions and oversampling.

The most prevalent (and simplistic) ray tracing technique, Whitted-style ray tracing, does not take into account many effects and phenomena we observe in nature [Whitted 1980]. A ray tracer that could accurately simulate every natural phenomena would be incredibly expensive to solve analytically.

*e-mail: mwilson3@umbc.edu

There have been several techniques for a more accurate analysis of the intensity equation. The one focused on in this paper is distributed ray tracing [Cook et al. 1984], which approximates the intensity equation by oversampling several things while ray tracing a scene:

1. The camera lens is oversampled, yielding a *depth of field* effect.
2. Shadows are oversampled, which results in softer, less rigid shadows, which are called *penumbras*.
3. Refraction and reflection rays are oversampled, resulting in softer versions of refraction and reflection. These soft versions are called *translucency* and *gloss* respectively.

This technique was chosen for its ease of understanding and relative simplicity, though most ray tracing techniques could be adapted for this purpose.

Typically, computer graphics in general are improved upon to more accurately reflect the natural world. Though it has been mentioned that at the present, a completely accurate ray tracer would be too computationally expensive to compute in a reasonable amount of time, improving realism via approximations can yield fairly accurate and visually appealing results. Additionally, approximations can cut computation time down considerably, which makes using them more attractive.

For this reason, the author of this paper has decided to add mirage rendering to a distributed ray tracer. In the introduction, it was mentioned that light bends through the air conforming to a parabolic curve when a temperature gradient exists in the air. A special object which we can refer to as a *mirage box* can be used to approximate this parabola. Its implementation will be described later in this paper.

2 Related Work

As was mentioned earlier, distributed ray tracing is a major component of this paper. Distributed ray tracing simply oversamples at several points in the rendering process, and yields more accurately ray traced images because of it [Cook et al. 1984]. Refer to the introduction section for a more thorough explanation.

Another technique that improves upon the realism of ray tracing is the metropolis light transport technique [Veach and Guibas 1997]. The MLT technique achieves very impressive results using a Monte Carlo algorithm that samples the paths from light sources to the lens. This method creates very convincing looking global illumination effects, which were not considered by this paper. The MLT technique was not selected for this paper due to its slightly more complicated math.

Mirage physics are explained well by Andrew Young [2006]. This article explains that mirages are not optical illusions, which is a common misconception. It discusses the bending of light through thermal gradients of air, and clarifies this with the term "atmospheric refraction."

Other attempts at ray tracing natural phenomena include ray tracing fog, clouds, flames, dust, and other types of particle effects [Kajiya and Herzen 1984]. This paper discusses the physics behind fog and

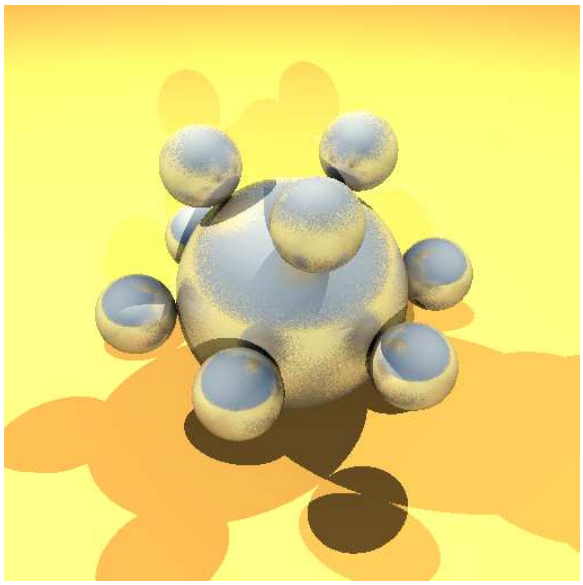

Figure 1: *Translucency and gloss are correctly modeled.*

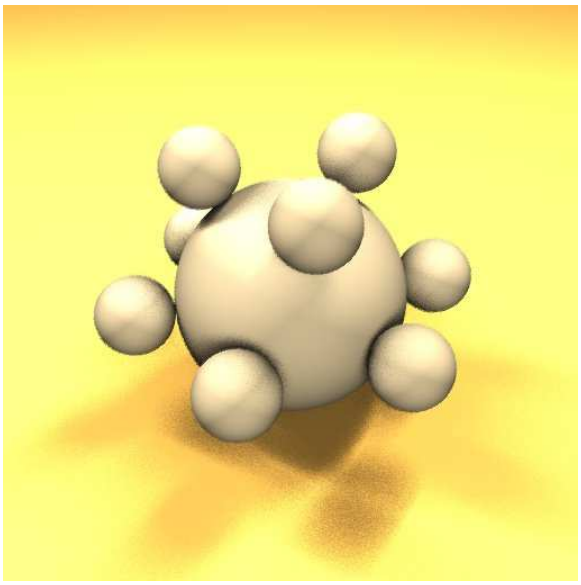

Figure 2: *Penumbras display correctly.*

mathematical equations associated with it. Kajiya and Herzen are able to solve these equations, which are too difficult to solve analytically, by making a number of assumptions that ease the computational complexity.

3 Implementation

The implementation of this ray tracer was tackled in two separate stages. First, the distributed ray tracer was implemented. Second, the mirage rendering capability was implemented. The NFF specification needed to be modified to support the new additions to the ray tracer as well.

3.1 Distributed ray tracer

The distributed ray tracing stage was relatively simple. There were four things that needed to be implemented that were listed earlier in this paper, though they will be listed here again for ease of reading:

1. Depth of field
2. Penumbras
3. Translucency
4. Gloss

Implementing all of these was a matter of modifying an existing ray tracer to handle these components.

In order to implement depth of field, the eye point was viewed as a lens of sorts. Random points were taken from the surface of the lens, and then each of these samples were then used as an "eye point," used normally in the ray tracing function, and averaged together. This produced acceptable depth of field type effects.

Penumbras were handled similarly. To produce them, a standard shadow ray was traced normally (standard shadows can be observed in figure 1). Then several other rays were distributed randomly around the shadow ray, all pointing in generally the same direction. These are averaged together, producing results like those seen in figure 2.

Finally, translucency and gloss could be handled similarly. A function was added that fired a ray off normally, then added sample rays around that direction, and averaged them together. You can see the results of this in figure 1.

There is a motion blur component to distributed ray tracing, which can be implemented by sampling at the same point over a series of frames. However, for the purposes of this project it was unnecessary to implement, as the author of this paper was not concerned with motion.

3.2 Mirage rendering

Mirage rendering was a relatively simple addition to this ray tracer. Mirages are created by light bending through the air in the shape of a parabolic curve [Wikipedia 2007]. While analytically solving this would add a lot of computational complexity to a ray tracer, the path can be approximated through the use of a multilayered mirage box [Berger et al. 1990].

Each layer of the mirage box is capable of refraction or reflection. When a ray enters the mirage box, it can be refracted and reflected in an approximately parabolic shape. This can create very convincing results.

This was implemented using a series of four-vertex polygons. The ray tracer used for this project already had a polygon rendering facility built in. Adding the mirage box was as simple assembling planes in a box type shape and making them all transparent and refractive.

4 Results

Several figures already present in the document show the efficacy of the distributed ray tracing part. Unfortunately, the distributed ray tracing component severely decreased the performance of the ray tracer. Instead of having a maximum of seven rays per pixel (as the ray tracer limited the number of ray bounces to seven), it was now possible to have upward of $samples^7$ per pixel, which is a significant increase. More generally, you will get $samples^{bounces}$ possible rays.

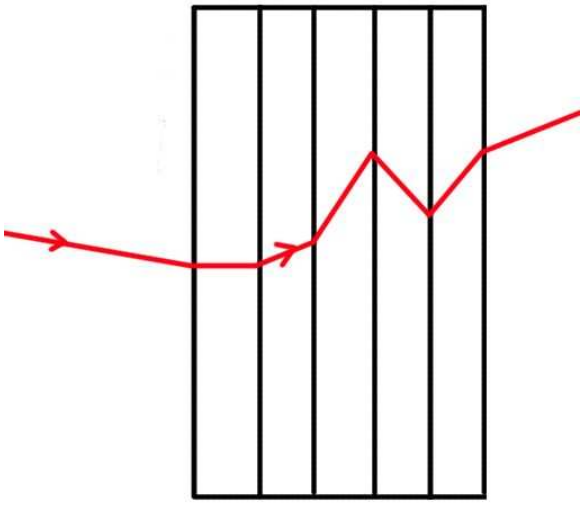

Figure 3: *Penumbras display correctly.*

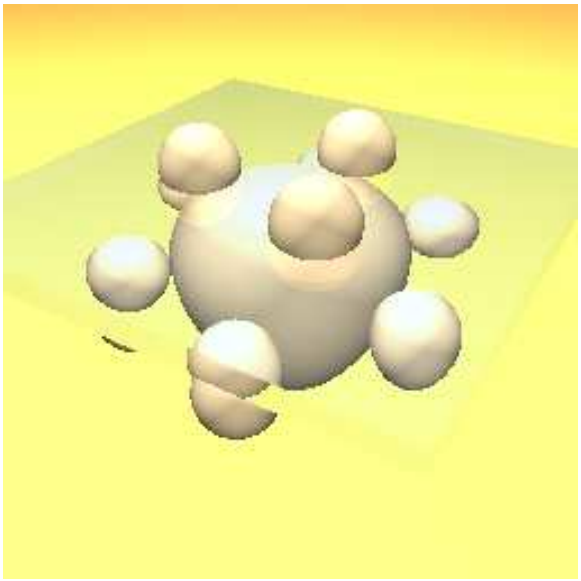

Figure 4: *A 2 layer mirage box on top of a ball model.*

The mirage box too decreased the performance of the ray tracer. When using a mirage box, the number of bounces necessary can become extremely large (greater than the five that the ray tracer was originally limited to). The number of bounces was increased to seven, as any larger generally decreased performance unreasonably. The 256x256 pixel picture in figure 4 took 38 minutes and 57 seconds to render on a AthlonXP 2400 with 1 GB of RAM running Ubuntu Linux 7.04. Samples were decreased to 1 for this rendering.

Due to a lack of resources, the author was unable to capture a picture of mirage ray tracing working with distributed ray tracing effects enabled. However, it is demonstrable that the mirage box works (see figure 4) and translucency works (see figure 5). There is no reason that both would not work in tandem given greater processing power.

Disregarding performance, however, one can see by looking at figure 4 that the mirage box does indeed work. The scene after having been refracted through the image box takes on a "mirage-like" appearance, which was the goal of this paper.

5 Future Work

There are several very obvious directions to take this paper in. The most obvious would be to speed up the ray tracer. This can be done several ways, via mathematical optimization of calculation, reduction of overhead, introduction of optimizing data structures (octrees, kd-trees), etc. Doing this would make it significantly easier to test.

Additionally, one could substitute another more accurate algorithm for distributed ray tracing. The metropolis light transport would make an interesting combination with mirage rendering due to the accuracy of its global illumination and lighting type effects.

Finally, adding other natural phenomena to this renderer would be a wonderful way to extend this paper. Mirages are very neat, but there are other things that could be added. Fog and global illumination are some phenomena that could be added, but there is certainly are many more that could be implemented.

6 Acknowledgments

The author would like to thank Dr. Marc Olano for his paper comments, as they helped me greatly. Also, thanks go to the author's CMSC 635 classmates, as their suggestions helped in more rapidly develop and test my application.

References

BERGER, M., TROUT, T., AND LEVIT, N. 1990. Ray tracing mirages. 36–41.

COOK, R. L., PORTER, T., AND CARPENTER, L. 1984. Distributed ray tracing. In *SIGGRAPH '84: Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 137–145.

DACHSBACHER, C., AND STAMMINGER, M. 2005. Reflective shadow maps. In *SI3D '05: Proceedings of the 2005 symposium on Interactive 3D graphics and games*, ACM Press, New York, NY, USA, 203–231.

HAINES, E., 1996. Neutral file format.

KAJIYA, J. T., AND HERZEN, B. P. V. 1984. Ray tracing volume densities. In *SIGGRAPH '84: Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 165–174.

VEACH, E., AND GUIBAS, L. J. 1997. Metropolis light transport. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 65–76.

WHITTED, T. 1980. An improved illumination model for shaded display. *Commun. ACM 23*, 6, 343–349.

WIKIPEDIA, 2007. Mirage — Wikipedia, the free encyclopedia, May. [Online; accessed 07-May-2007].

YOUNG, A., 2006. Mirages and green flashes. [Online, accessed 07-May-2007].

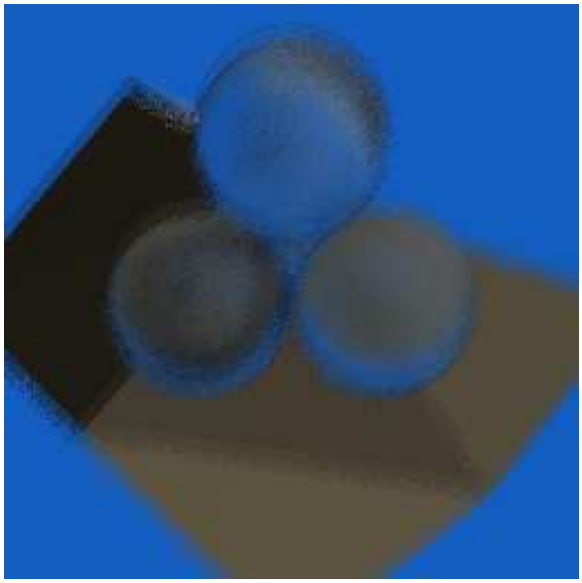

Figure 5: *Translucency and depth of field..*