Adding Realistic Camera Effects to the Computer Graphics Camera Model

Ryan Baltazar

May 4, 2012

1 Introduction

The camera model traditionally used in computer graphics is based on the "camera obscura" or pinhole camera. In this model, all light enters the camera through a small hole, the camera point, and is projected onto the rear image plane [5]. This produces an accurate representation of the objects in front of the camera, albeit upside down.

The graphics model in ray tracing follows a similar model. Like the pinhole camera, all rays must pass through a single point to strike the image plane. Unlike the pinhole, the image plane exists *in front* of the camera point rather than behind. To determine the color of a pixel, rays are shot from the camera point through the image plane into the scene where they can collide with objects. Figure 1 shows this process.

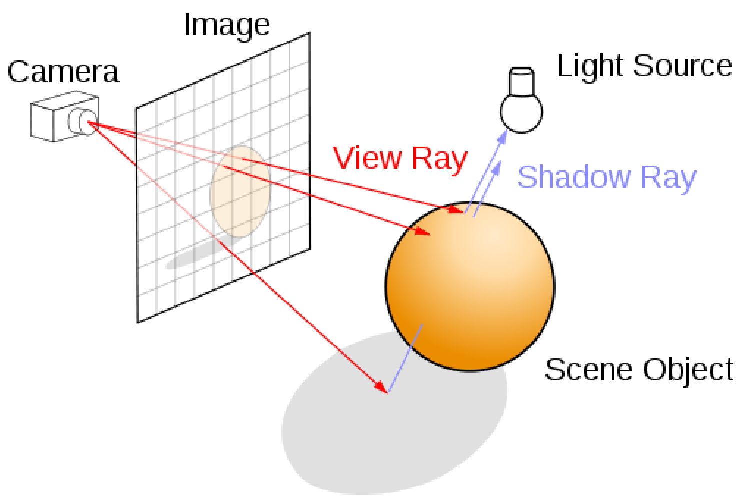

Figure 1: Rays traced with the traditional model are shot through the image plane out into the scene.

Unfortunately, this process has some undesired side effects. Most notably, it produces images which are perfectly sharp and in focus. While this can be helpful, it is usually more artistically pleasing to have the intended subject of the scene in focus while the rest of the image is fuzzy and out of focus. Another problem with this model is that there is no support for lens effects such as zoom or warping.

In this paper, I describe several techniques for generating what is known as depth of field, as well as for adding support for ray tracing through a system of lenses.

2 Related Work

This is certainly not a new problem, and there are several papers that describe techniques for solving these issues. Cook et al. [3] describe a method known as distributed ray tracing, which can be used to generate different effects in images. One of the techniques described is a way of using distributed ray tracing to simulate the depth of field effect.

Kolb [1] and Barsky [2] present similar methods enhacing the distributed model described by Cook to add effects from using lenses. Kolb also describes the necessary algorithms and storage methods for tracing a ray through a system of lenses.

3 Depth of Field

Depth of field refers to how in focus an image is. A scene is said to have a shallow depth of field, or shallow focus, if there is only a small portion of the image that is clear and focused. Similarly, an image has a deep depth of field, or deep focus, if most of the image is in focus. Having a shallow depth of field can help contrast the subject of the scene with the background or foreground.

The primary factor in determining depth of field is the camera's aperture. The aperture refers to how large the physical stop is before striking the camera sensor. Larger apertures produce a deeper depth of field; smaller apertures just the opposite.

3.1 Implementation

The implementation of depth of field in this project is a distributed model, similar to the one described in [3]. The user can specify the plane he or she wishes to focus on by using the up and down arrow keys: the up arrow key moves the plane further away, while the down arrow key moves it closer. A visualization of how far the focal plane is can be toggled using the 'd' key. The user also specifies how large the aperture is using the "-ap_size" switch on the command line.

The algorithm for determining the color of a pixel (i, j) is fairly simple. We first shoot a ray from the eye into the scene and determine where it intersects the focal plane. Then many rays are generated around (i, j) in the image plane and shot in the direction of this focal point. The average color of all these rays is the color of that pixel. For pseudo-code, see Algorithm 1.

To determine the intersection of the ray from the image plane to the focal plane, an actual ray trace is not necessary. Similar triangles can be used to determine the distance along the line to the focal plane, which can then be evaluated to give the point.

Algorithm 1 Calculate the color of a pixel (i, j)

```
$P_0 \leftarrow$ camera point.
$f \leftarrow$ focal plane distance
$P_i \leftarrow$ point in worldspace of $(i, j)$
$P_f \leftarrow$ IntersectFocalPlane($f$,Ray($P_0, (P_i - P_0)$))
$Color = \langle 0, 0, 0 \rangle$
for $i = 1 \rightarrow 10$ do
    $P \leftarrow$ random point in radius $\frac{\text{aperture}}{2}$ around $P_i$
    $R \leftarrow$ Ray($P_i, (P_f - P_i)$)
    $Color += $ TraceRay($R$)
end for
$Color = Color/N$
return $Color$
```

3.2 Sampling

Generating an image with depth of field is expensive. Instead of having just one ray go through the scene, it takes many rays. Not having enough rays produces an image that is grainy and is not pretty to look at. Getting even a decent result out of the depth of field ray trace needs at least 50 rays or more depending on how large the aperture is. Larger aperture diameters require more rays and vice versa. Figure 2 shows the effect of having different numbers of depth of field rays.

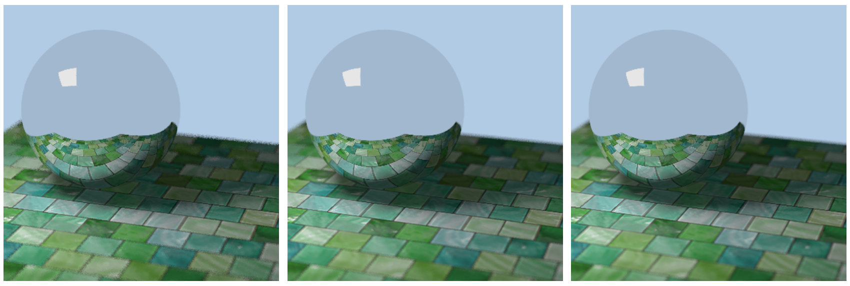
Figure 2: Left to right: 10 rays, 50 rays, and 100 rays.

To minimize the number of rays needed to produce a soft focus effect, stratified sampling could be used. However, it is not implemented here.

3.3 Results

The results of this implementation produce the expected sharp/blur effect which can vary with different apertures. The images in figure 3 show a scene with three mirror balls, blue, red, and green, coming in and out of focus as the plane moves.

4 Lenses and Ray Tracing

To generate effects from using lenses, I use a method similar to the one described in [1]. Rays are traced through a lens and then sent into the scene to determine

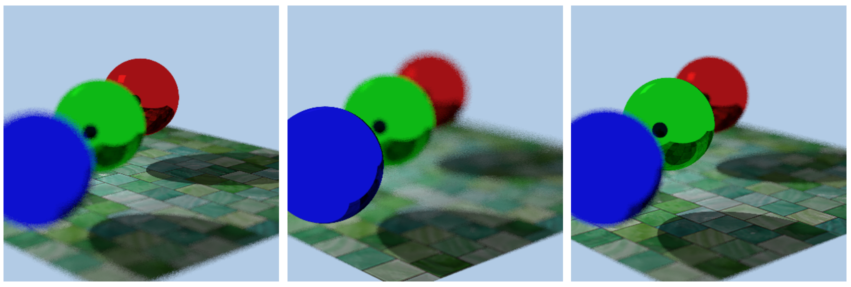

Figure 3: Each of the colored glass balls comes in and out of focus.

the ray's color. It is not enough to simply have any ray that hits a lens change direction towards the focal point because light that does not move parallel to a lens's primary axis behaves differently. In addition, lens systems have too many elements for this technique to be applied.

4.1 Lens Representation

Lenses are represented as a series of surfaces from object space to image space. Each lens surface, or element, has a signed radius of curvature, thickness, index of refraction, and aperture. The radius of curvature refers to the radius of the sphere that the surface is a slice of. The sign is necessary for telling whether the surface is convex or concave when viewed from in front of the front lens (object space). A positive radius of curvature indicates that the element is convex when viewed from the front. A negative radius of curvature indicates that it is concave. If the radius of curvature is zero, then the surface is assumed to be planar. The thickness refers to the distance along the primary axis between this element and the next. The index of refraction refers to the index of refraction that a ray will enter when passing through the surface. A missing index of refraction signifies that the ray will enter air after moving through the surface, which has index of refraction of $n_d \approx 1$. Finally, the aperture refers to the diameter of the element. Rays outside of the aperture are said to be blocked and return black as their color.

Lenses are stored in a plain text file that is tab delimited and supports curved elements and planar stops. The camera keeps a list of all lens elements read from this file from object to image space and uses this list to trace rays.

4.2 Focal Distance

For a simple lens, it is often helpful to know the focal distance of the lens. Using Snell's law and some math, we can derive the Lensmaker's Equation for a simple lens:

$$\frac{1}{f} = (n-1)\left[\frac{1}{R_1} - \frac{1}{R_2} + \frac{(n-1)d}{nR_1R_2}\right] \tag{1}$$

where n refers to the index of refraction of the lens, d is the distance between the two surfaces along the primary axis, n is the index of refraction, and R_1, R_2 refer to the radius of curvature of the two surfaces. It is important to note that signs for R_1 and R_2 are significant. R_1 is convex if it is positive and concave if it is negative. R_2's signs are reversed when looking from object space: it is positive if concave and negative if convex.

4.3 Lens Intersection

To intersect a given ray with a lens element, we simply intersect it with either a plane or a sphere, depending on the type. If the element is planar, then the intersection between the ray parameterized by t and the plane is given by evaluating $R(t)$ at t' where

$$t' = \frac{-(D + n \cdot R_0)}{n \cdot R_d} \tag{2}$$

The distance to the aperture is merely the length of the vector between this point and the primary axis.

If the element is spherica, we intersect it with a sphere with a sphere centered at a point given by

$$c = P_c + \text{thickness so far} - \text{element radius} \tag{3}$$

which has the given radius of the element. To then determine the perpendicular distance between this point and the primary axis, we simply apply the following formula:

$$d = |x(t') - x_0| \tag{4}$$

where x_0 is the point of intersection and $x(t)$ is the line that describes the primary axis: $x(t) = P_0 + rt$. It can be shown that

$$t' = (x_0 - P_0) \cdot r \tag{5}$$

For either case, if the distance is shown to be greater than the aperture, then the ray is blocked.

4.4 Ray Tracing Through Lenses

To trace rays through the lens systems, I implemented two methods. The first method enhanced the traditional method by placing a lens in front of the camera point and tracing the single ray through the system. The second method shoots many rays at the back element and averages the return color to get the final color of the pixel.

In both cases, tracing a single ray through the lens system follows the same algorithm. Given a ray R, intersect it with each element back to front. If there is an intersection and the ray is not blocked, refract the ray if necessary and update R.

5 Results

The results from tracing rays through the set of lenses was mostly successful. Figure 4 shows the results from shooting approximately 100 rays at the back of a lens. Figure 5 shows the results from shooting approximately 500 rays at the

Algorithm 2 Trace a ray R through the system

```
$R$ ←Ray(Point on image plane, Direction)
for Each lens element $L$ back to front do
    if $R$ does not intersect $L$ then
        return blocked
    end if
    if Index of refraction on far side of $L \neq$ current index of refraction then
        Update R using Snell's Law
    end if
end for
return $R$
```

back of a 50mm double-gauss lens with a 2mm aperture. We can see that not all the rays make it; in fact, a significant subset of the original number actually pass through.

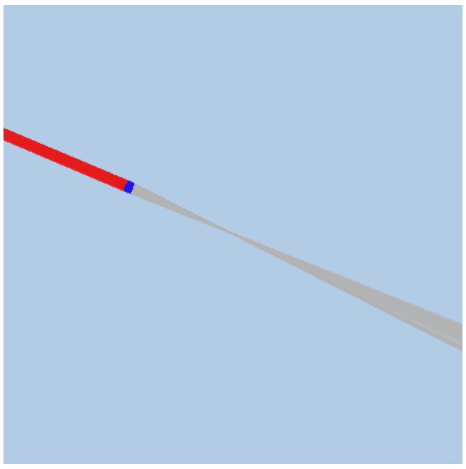

Figure 4: Rays converging on a simple lens with both elements radius of curvature 80mm.

Figures 6 and 7 show examples of two different lenses from the same camera positions and their respective renderings. The images are upside down because the light is being bent from the lens, as well as the fact that the image plane behind the camera point.

5.0.1 Limitations

The scattering method is used to imitate depth of field. By varying the aperture size of the element in the 50mm double-Gauss lens, we change how blurry the image is. Unfortunately, without a way to focus the lens, I was unable to get a clear render of the scene.

Another problem is that it takes a very high number of rays to produce an image without black spots. Since not every ray makes it through the system, if we can determine a region in which we should trace rays, we can significantly cut down on the number of rays needed. In particular, the most limiting element

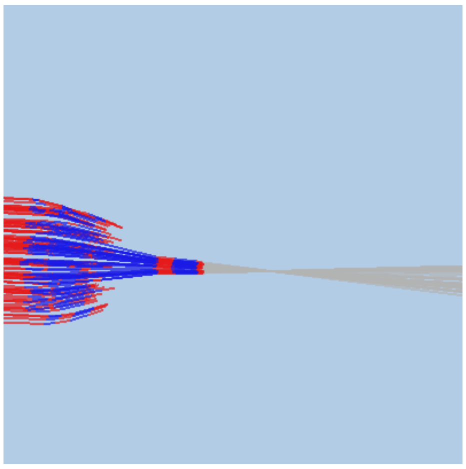
Figure 5: Rays passing through the 50mm double-Gauss lens with a 2mm aperture

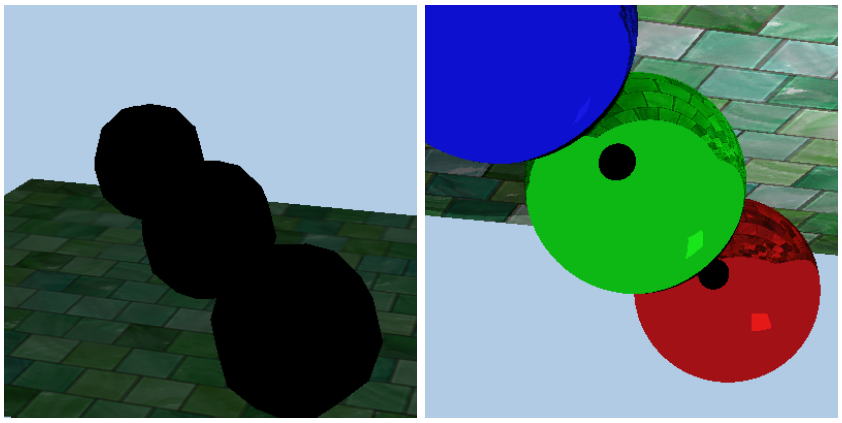
Figure 6: A simple lens with a 500mm radius of curvature is placed in front of the lens with the camera position on the left to produce the image on the right

is the aperture of the lens, which exists somewhere in the system. Determining what is known as the exit pupil — the image of the aperture through the lenses —would greatly help the efficiency of the ray tracer. This is because every ray that is sent through the exit pupil is guaranteed to make it through, rather than simply shooting hundreds of rays at the back-most element and hoping to get lucky.

6 Future Work

There are many directions I would like to continue to take this project. In particular, I want to continue studying lenses and lens systems to figure out a method of focusing a lens instead of rendering just a blurry mess. This would lead to some interesting images, which would not only look good, but be phys-

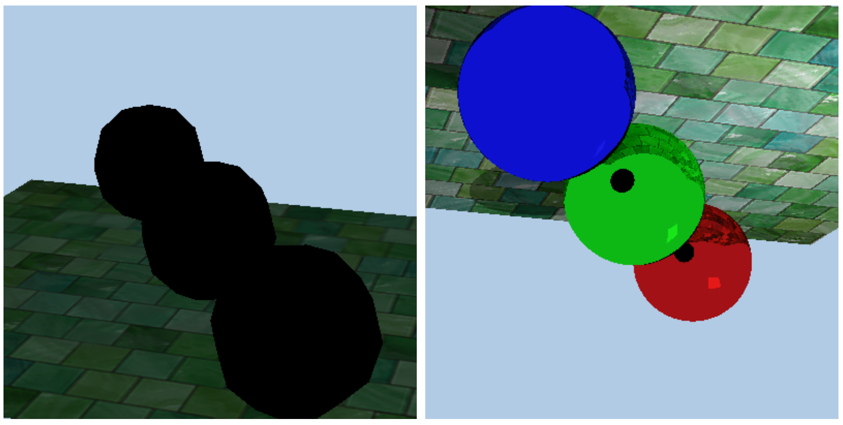

Figure 7: A simple lens with a 70mm radius of curvature is placed in front of the lens with the camera position on the left to produce the image on the right

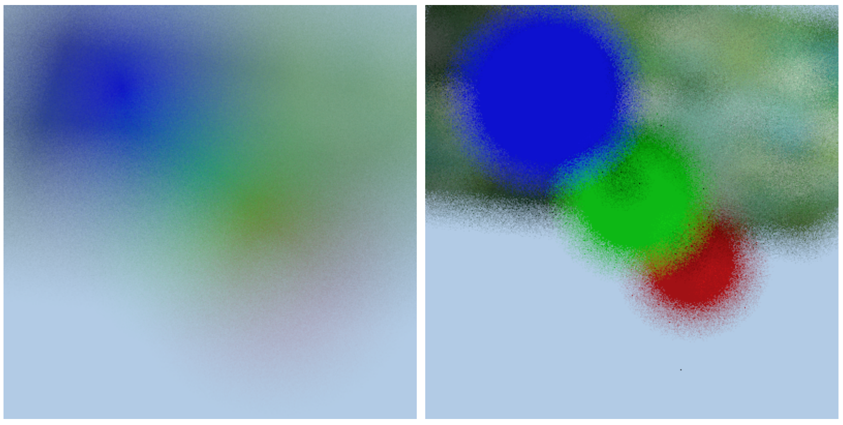

Figure 8: Left: aperture size of 10mm. Right: aperture size of 2.0mm

ically accurate.

Additionally, the running time could be greatly improved by applying stratified sampling to the initial depth of field implementation. It would take fewer rays to produce a high quality image, rather than simply randomly generating rays and hoping the spread is completely uniform for the sample.

As mentioned before, determining the exit pupil would also improve the running time. This would help when the lens has a small aperture because it means that fewer of the rays would go to waste being blocked, saving precious CPU cycles. While it is not a major problem with large aperture sizes, small aperture sizes (e.g. about a millimeter) would greatly benefit from this improvement.

References

[1] Craig Kolb, Don Mitchell, and Pat Hanrahan. A realistic Camera Model for Computer Graphics. In Robert L. Cook, editor, ACM SIGGRAPH 1995

Conference Proceedings, pages 317–324, Los Angeles, August 6–11 1995.

[2] Brian A. Barsky, Daniel R. Horn1, Stanley A. Klein, Jeffrey A. Pang1, and Meng Yu. Camera Models and Optical Systems Used in Computer Graphics: Part I, Object-Based Techniques. In Proceedings of the 2003 International Conference on Computational Science and its Applications.

[3] Robert L. Cook, Thomas Porter, Loren Carpenter. Distributed Ray Tracing. In Proceedings of ACM SIGGRAPH 1984.

[4] Fowles, Grant R. Introduction to Modern Optics. New York: Holt, Rinehart and Winston, 1968.

[5] Johnson, Charles S. Science for the Curious Photographer: An Introduction to the Science of Photography. Natick, MA: A.K. Peters, 2010.