UNIVERSITY OF CALIFORNIA, SAN DIEGO

Diffraction Gratings in Ray Tracing

A Thesis submitted in partial satisfaction of the requirements
for the degree Master of Science

in

Computer Science

by

Andre James Watson

Committee in charge:

Professor Henrik Jensen, Chair
Professor Mattias Zwicker
Professor Sam Buss

2009

The Thesis of Andre James Watson is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

Chair

University of California, San Diego

2009

DEDICATION

Thank you to UCSD for providing me an interesting learning experience. But, most importantly, thank you giving me the credibility to go anywhere in the computer science world.

TABLE OF CONTENTS

LIST OF FIGURES

ABSTRACT OF THE THESIS

Diffraction Gratings in Ray Tracing

by

Andre James Watson

Master of Science in Computer Science

University of California, San Diego, 2009

Professor Henrik Jensen, Chair

Diffraction has been relatively untapped so far in the rendering environment. By making some tweaks to an existing renderer, specifically its light handling ability, an effective diffraction grating can be simulated and used. Previous approaches to the problem have been computational heavy or produced unrealistic results, and all of them were focused solely on simulating optical disks. This approach creates a usable ray tracing component that allows for general diffraction, which can be applied in numerous ways, including optical media.

Introduction

Computer generated images have started to approach a level where they are almost indistinguishable from real photos. This has led to their use in both movies and video games, sometimes with the observer not even realizing they are looking at a produced image rather than a recording of reality. This trend towards realism continues and has always been desired in the arena of graphics research. It has come such a far way that some of the finer details of a rendered scene are being looked at. One of these details is diffraction.

While there are not a great number of naturally occurring examples of diffraction, the effect is nonetheless something that does appear. Most notable, the data side of a CD or DVD is an example of diffraction and something that almost everyone encounters on a daily basis.

So far, only a few attempts have been made at trying to reproduce diffraction in any sort of graphical environment. As presented later, these attempts all contain certain drawbacks. Either they are too computationally intensive for practical use, they do not look very realistic in the end, or they rely too heavily on one specific environment for their use. One common drawback to all the previous work is that it is all focused specifically on reproducing a CD or DVD. There is no attempt as simply doing general diffraction as any physics student would see in their introductory classes.

I propose that diffraction can be done using basic methods of ray tracing, and done without a huge computational overhead. By applying this kind of methodology, diffraction could be easily loaded into almost any renderer, and be used for surfaces other than optical disks.

Background Theory

The research paper for light diffraction was originally published in 1665 from the work of Francesco Maria Grimaldi. In 1803, Thomas Young conducted experiments and found that the only way he could explain his results was to interpret light as having wavelike properties. A few years later, Augustin-Jean Fresnel also conducted diffraction research and further argued towards the wave theory of light. This wave theory of light has long since been considered fact among the physics community and is the main reason for the observable diffraction effect.

Diffraction is caused when light is incident upon a very small opening whose width is smaller than the wavelength of the given light. This opening is usually referred to as a "slit". The light exiting from the other side of this slit emits in all directions in a half circular pattern. This pattern would be analogous to watching water ripple when something is dropped into it. The water ripples propagate away from the perturbation point and carry the impact energy away in all directions. This is exactly what is happening through a diffraction slit, except that the waves only emit in a half circle as opposed to a full circle, and the phase of the light is determined by the distance from the slit, whereas the water phase is determined by time.

The description so far would imply that after going through a slit, light is visible in all directions on the other side, but this is not the case. There is more is at work here. Because light is wavelike, and the slit has to have a width in order to allow light to pass through, the width of the slit causes the slit itself to be the source of multiple starting waves. Thing about this as if you were pouring water into a funnel.

Wherever you start pouring the water into the funnel, the water travels into the bottom, but if you pour it in a different spot, it comes out of the bottom slightly different but no less on the same course. The same is true for the slit, in which case the light incident more the left emits slightly differently than light that is incident more the right. To see a visible explanation of this, see Figure 1 where each wavelength along the aperture is emitting its own wave.

This effect of multiple emissions, even from one single slit, creates an interference situation. Imagine dropping to rocks into a body of water at a distance from each other. Each rock would create ripples in the surface that traveled outwards in all directions. But, where these ripples started to cross, you would begin to see interference. Wherever two high points in the ripples happened to be, the wave would be twice as high. Wherever two low points in the ripples met, there would be a large dip. And, wherever and high point from one ripple met a low point from the other ripple, the water would appear unperturbed. This is exactly the type of effect we can visibly see from diffraction. In Figure 1, you can see where certain angles of the diffracted emission appear normally waved, but other areas become grayed out as different phases of the emitted light from the different sources in the aperture effectively cancel each other out.

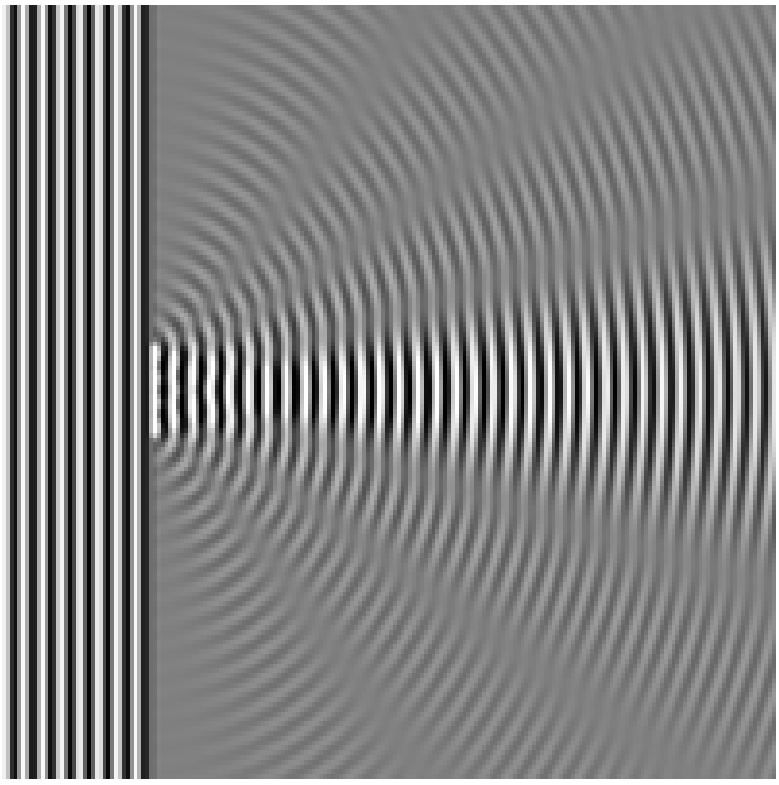


Figure 1: A four wavelength wide slit [13]. Each wavelength along the slit emits its own light waves, creating points of interference as the waves propagate outwards. The black and white represent the up phase and down phase of the emitted light wave. This is constant for a given position as opposed to a water ripple which oscillates. The grayed areas represent a region where an up phase from one emitting point combines with the down phase from another emitting point, canceling out the emitted light.

On the emitting side of a slit, there appear bright bands of light that fade out to no light at all, and then reappear again to form another bright band. This effect comes from having multiple maxima and minima at different angles relative to the slit

normal. The effect creates full construction and destruction interference, so the destructive points literally show no visible sign of any incident light, whereas the constructive points fully show the light incident the slit.

Because the phase of light is determined by distance and is constant for a given distance, the aperture of the slit determines where the interference angles of the light exist. By taking the angle relative to the normal of the slit, the amount of interference can be calculated and the intensity of light for that angle can be found.

The base case for diffraction comes from single slit diffraction. This type of slit contains one vertical slit and will be discussed more in the next section. This "single slit" experiment is the basis of the diffraction concept. Even when more slits are added and more interference points are caused, the maxima will always fall within the maxima of the single slit. This effect can be seen in Figure 2.

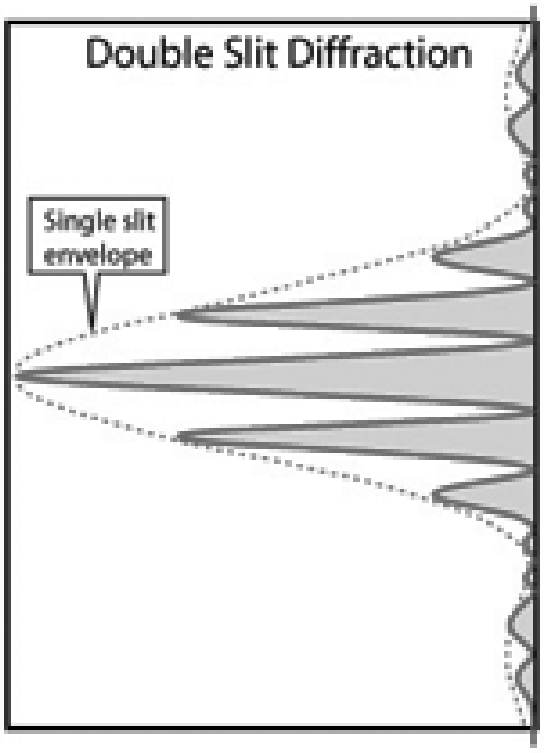


Figure 2: Double Slit maxima will fall within the maxima of a single pattern and have minima in the same locations as well [14].

This is also why CDs create the diffraction pattern. The data burned into the surface of the disk creates a groves in what was originally a shiny, smooth, metallic surface. These burned tracks of data are written circularly at even intervals around the disk. This means there are bands of very small reflective grooves right next to each other. This creates the diffraction effect when light shines on the CD. Note though, that the data is burned into the CD in a digital fashion. This means that not all parts of the data track are burned into the surface of the disk. That is why the CD still appears

very reflective like a mirror, even though diffraction is occurring because of the data rings.

It should also be very carefully recognized that the diffraction process is wavelength dependant. Each wavelength has, obviously, a different wavelength. This means the effect of the aperture is different relative to each spectrum of light. This will essentially give a prism effect any time any time white light is incident upon a slit. This is also why CDs appear to have a large hue of colors. This also changes where the constructive and destructive interference regions occur in the output pattern. Therefore, the area that red light constructively interferes will be at a different point than where blue light constructively interferes.

The other important thing to note is that it is assumed that the light coming into the diffraction slit is coherent. That is, that the light coming into the slit is in-phase with itself. If the light coming into the slit has interference going on already, the behavior of the slit is undefined as far as I can tell. I was unable to find any information regarding what happens in this case.

For the purposes of everyday, normal situations, the incoming light is essentially coherent with itself. This is why CDs and things like butterfly wings appear so iridescent under light.

Besides the angle from the slit, the shape of the slit is also important in predicting the visible diffraction pattern. The most common type of slit usually encountered is a vertical slit or a series of vertical slits called a grating. This type of slit is essentially a one-dimensional slit because it has infinite height, relative to the wavelength of the incident light anyways, but a narrow width aperture that causes the

diffraction. The pattern that emerges from this is also one dimensional in change and creates a band of light that shows the diffraction pattern. An example of this can be seen in Figure 3 on both of the left side images.

When the slit becomes narrow enough to cause diffraction in two dimensions, other interesting patterns emerge. A square pattern, for instance, creates a cross of sorts with some other patterning in between the arms. Circular patterns emit something that looks exactly like a water ripple frozen in time. A hexagonal pattern creates six spokes of maxima lines. All of these can been seen in Figure 3.

For the purposes of a CD, it is essentially a circular grating of vertical slits. What I mean by this is that the slits on the CD are really just a series of many veritical slits. Any slice taken out of the CD from the center along the radius is really a series of straight veridical slits that compose a grating. However, when looked at on the whole, they appear to be circularly oriented. This means the CD can be treated as a much simpler problem by looked at it on the small scale.

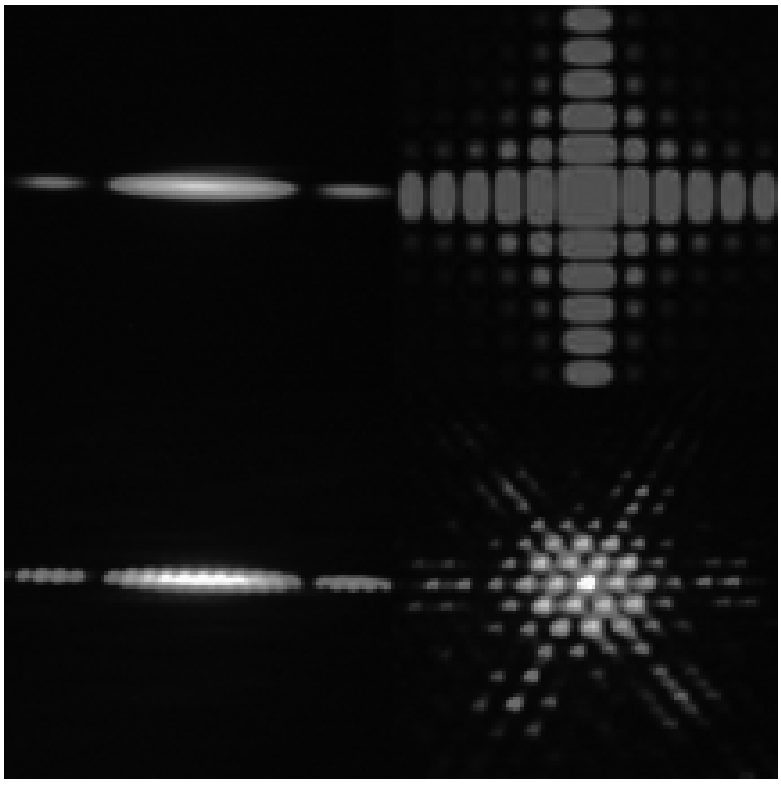


Figure 3: Types of slits [15]. Top-left: Single vertical slit. Bottom-left: Double vertical slit. Top-right: Square slit. Bottom-right: Hexagonal Slit.

Experimentally, the minima and maxima locations have been known almost since the beginning of diffraction research. These can be derived relatively easily simply by looking at the pattern produced by a single slit experiment.

The basis for the maxima comes primary from determining the phase angle from the given angle relative to the slit normal. The phase angle δ is dependant on the light wavelength and is displayed by the following equation where *a* is the aperture of the slit, θ is the angle of the light incident upon the resulting surface from the diffraction grating normal, and λ is the wavelength of the light. What this really

shows is the superposition any light waves will be in when they travels in the direction specified by θ. This is directly related to the aperture width *a.*

$$\delta = \frac{2\pi a \sin\theta}{\lambda}$$

After determining the phase angle δ, the resulting intensity along that angle θ is determined by the following equation where I_0 is the incident light upon the slit.

$$I = I_0 \frac{\sin^2\left(\frac{\delta}{2}\right)}{\left(\frac{\delta}{2}\right)^2}$$

This intensity equation only works, however, for a single slit. Because a multiple slit grating is actually composed of a bunch of single slits, the equation needs another term to describe the phase shift, or the interference from all combined slits, between the slits themselves. This is done by taking the phase angle of the slits relative to each other.

$$\xi = \frac{2\pi b \sin\theta}{\lambda}$$

The phase angle equation looks exactly like the one above, except that this time *b* is the distance between the slits. Let's call this new phase angle ξ to avoid confusion. The intensity for multiple slits interfering with each other look likes the single slit equation, except it account for the number of slits N. The intensity for N slits then becomes:

$$I = I_0 \frac{\sin^2\left(N\frac{\xi}{2}\right)}{\sin^2\left(\frac{\xi}{2}\right)}$$

What this equation shows is essentially how the different slits interfere with each other's output. But, this does not take into account the individual slits which affect I_0 already. Each individual slit has its own constructive and destructive interference. To account for this, we multiple the new intensity equation by the interference pattern of a single slit. Adding in the individual slits gives the overall multiple slit intensity equation.

$$I = I_0 \frac{\sin^2\left(\frac{\delta}{2}\right)\sin^2\left(N\frac{\xi}{2}\right)}{\left(\frac{\delta}{2}\right)^2\sin^2\left(\frac{\xi}{2}\right)}$$

This equation solves for the majority of cases I will be testing in this paper. Note that this equation only works in the case that *a* and *b*, the widths of the apertures and distance between the apertures respectively, remains constant.

At the present time, I must note that I have had some difficulty in finding accurate descriptions of the intensity patterns caused by any other two dimensional slit besides the circular slit. While it would be straight forward to just run a simulation, this is impractical from the approach that I will be starting from, though it could be expanded to use from the ideas presented in this paper.

Related Work

Surprisingly little work has been done in the realm of diffraction in a graphics environment of any kind. Some of the main works so far in this area I will discuss in this section.

Diffraction Shaders was one of the first attempts at simulating diffraction and was a paper that took a rather unique approach to the problem. Instead of relying on classical light dynamics for their calculation methods, they instead turned to quantum mechanics specifically to be able to exploit the wave nature of light to create an accurate model. The results were appealing, but I do not believe they were very realistic, and the algorithm has a couple of drawbacks. Firstly, using quantum mechanics and Fourier transforms is incredibly expensive. It is understood that ray tracing is, of course, an expensive process, but throwing multiple layers of integrals, transforms, and partial differentials into the process is even more expensive. If most rendering environments were physical simulations, this would be very appropriate. But, for most normal graphics uses, this is almost certainly overkill. Their assertion that they were able to implement their algorithm in Maya proves the algorithm is worth its expense is a bit of a stretch to claim considering the simplicity of their test scenarios versus most rendering contexts. This is further seen by their use of Rayleigh scattering in their approach. While this effect is definitely part of the physical world and is used to explain why the sky is blue, this effect is virtually unseen in most day to day scenarios. Let alone, the sky is usually better simulated with participating

medium algorithms. Again, this comes down to the fact most environments are not intended to *be* physically realistic but to *look* physically realistic. Another interesting aspect of this approach is the disappearance of the Fresnel term. It would seem that this should remain despite the anisotropic argument they make. During the time period of this paper, it may very well be true that most views of the physical world were isotropic, but this is not the case anymore. Anisotropic surfaces are seen in many places and produce very nice results now. I will come back to their anisotropic view shortly. Their approach to use quantum mechanics to have access to the wavelike properties of light is definitely a novel approach. Diffraction is based off of the interfering of light waves, and they do a good job of differentiating interference versus diffraction. Though, it should be noted, the two essentially are very different versions of the same effect. Both are superimposing wave values onto each other, but the effect is seen different in both cases. For this reason, I would argue that using quantum mechanics to simulate the diffraction effect is completely unnecessary, only because the interference effect can be simulated without trouble using classical methods. That said, they model diffraction through the modern physics approach. However, they tend to focus on the anisotropic nature of the surface to produce the effect. I feel this is somewhat misleading, because the surface itself is not really causing the effect. As stated before, the effect is created by the emission of light in all directions from the diffraction grating. Their model assumes that only certain light reflects in a very standard BRDF kind of method. This is certainly a novel approach, but it ignores the real core of what is happening at the heart of the diffraction equations. That said, their simulating of the metal brush surface looks amazing, as seen in Figure 4. The CD

from Figure 5, however, appears distorted from a real CD, mainly in that their color bands appear too wide. I would attribute this to their surface based approach to the problem.

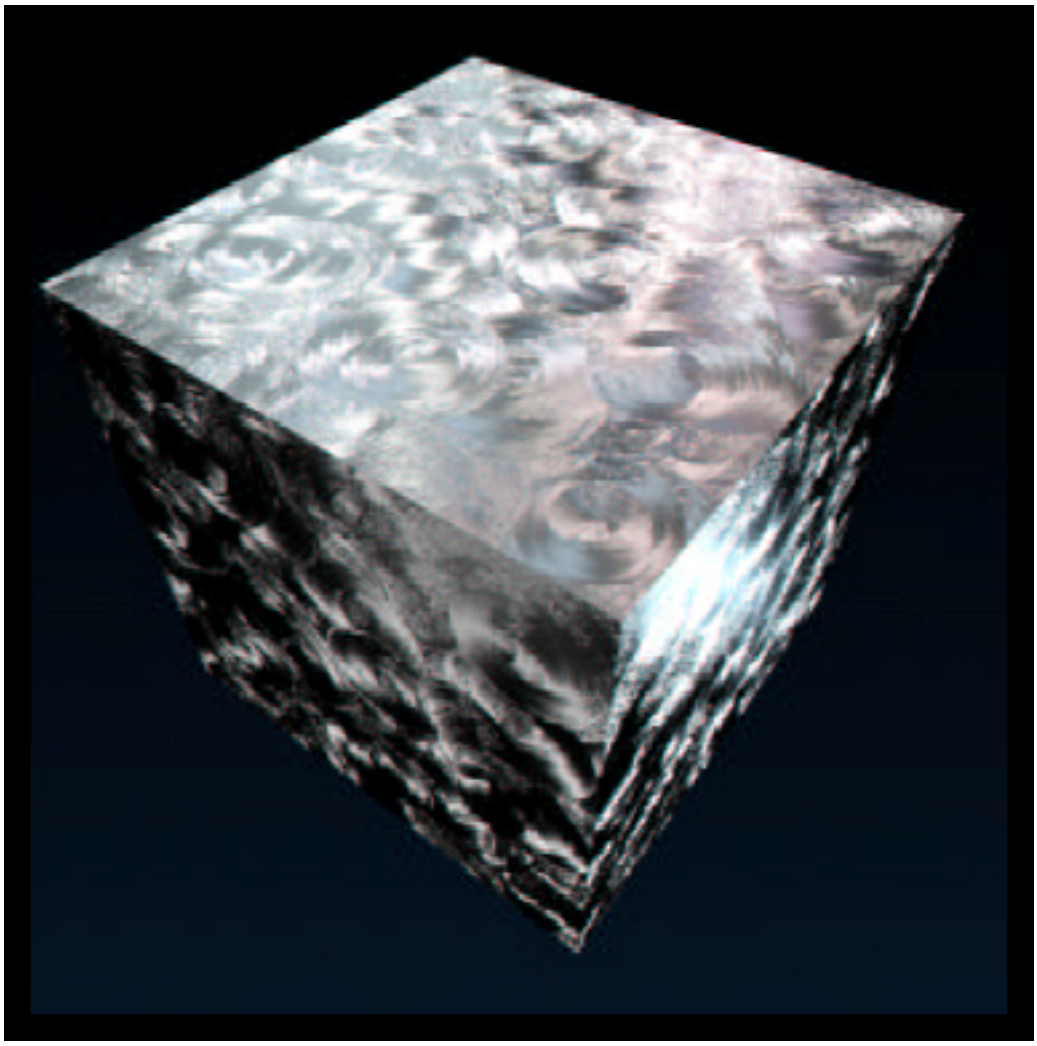


Figure 4: The brushed metal cube from Diffraction Shaders.

Figure 5: The rendered CD from Diffraction Shaders.

Not long after Diffraction Shaders came Tsingo's geometric approach to rendering diffraction. Mainly, this paper was commentary to the popular opinion, at the time, that rendering diffraction was not possible in a traditional ray tracing environment. This opinion persisted because modeling the wavelike properties of light to achieve certain modern physics effects either requires modeling the quantum nature of light which is vastly complicated, or it requires some clever approach to remove the underlying complication of the quantum mechanics involved. No such approach to the latter existed yet, so the opinion was widely true. This was also

further complicated by the fact that all existing regular ray tracing methods were heavily relying upon BRDFs to create their surface properties. Trying to create a BRDF to model a modern physics effect could be difficult, if not impossible. To show that a regular ray tracing environment could be used to model diffraction, Tsingo's approach relied mainly upon the Geometrical Theory of Diffraction. This allowed him to avoid using modern physics equations to solve for the wavelike qualities of light. This, however, was not enough for a complete solution to model diffraction and still required some major assumptions about the surface to allow the diffraction simulation. His novel approach was to treat the surface of a CD as a series of pointy wedges sticking up off the surface. This makes sense as a place to start because as stated in the paper, modeling interference and diffraction takes places a very small scale, much smaller that a typical ray tracing environment takes into effect. Because of this, certain assumptions need to be made as to the nature of the surface in a way that can be calculated without forcing the ray tracer to get down to wavelength wide distances in its calculations. He even mentions how some approaches sought to use Snell's law and carry phase information with the traveling light in the scene, which almost sounds a little similar to photon mapping. This approach is almost a hybrid to using quantum mechanics and ray tracing, but again complicates the calculations considerably because normal ray tracing does not operate on this small scale. For these reasons, Tsingo assumes wedges for the surface of the CD. However, this would appear on the surface to essentially suffer the same argument of scale scoping. I seriously question the wedge approach not because it does not model the problem, but because it does not accurately model the problem. A CD surface is flat and contains and large number of

square grooves. To treat these grooves at point wedges is automatically assuming the surface is something other than what is really is. Which, combined with the real physics being used to simulate the diffraction effect, seems to hint that the simulation will already not appear as it should. This is, however, a good start to the problem. This statement can be seen very clearly when the results are viewed.

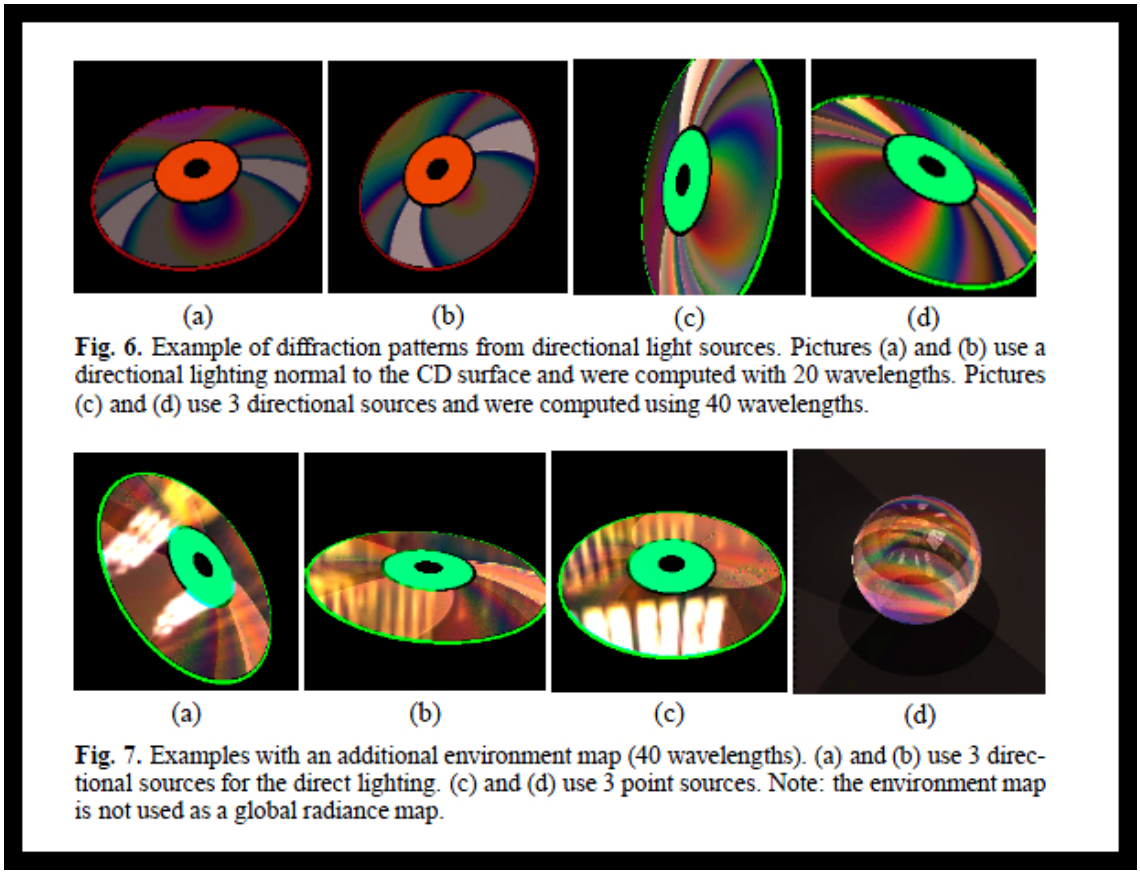


Figure 6: Tsingo's rendered CD.

The CD, as seen in Figure 6, looks very unrealistic. Whereas Diffraction Shaders had somewhat wide color bands, Tsingo has even wider bands; so much that

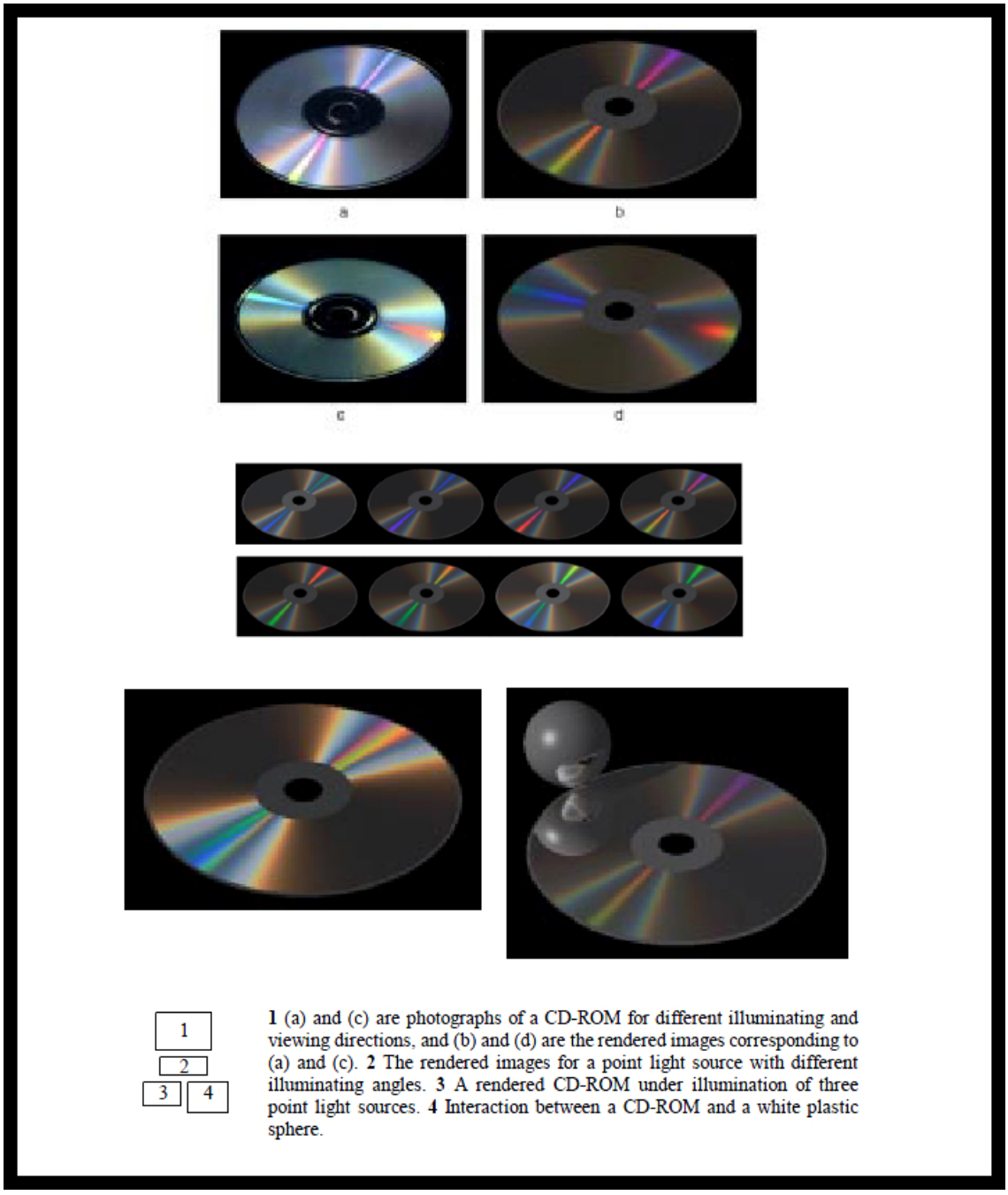
the color spectrum disperses across the entire CD surface. This is definitely not how a CD appears in the everyday world. The other major problem with this approach is the calculation time involved. For the small images in the results section, the calculation is astronomical compared to a normal scene. If the scene were very complicated the calculation would be exponential a normal ray tracing environment. Most of this stems from the problem of finding the shortest path from a point through the CD surface to a light source and then back calculating the diffraction involved. The assumptions that go into this are novel, especially in their ability to a reduce an NP-complete problem down to one that can be solved with relative speed, but it is still a huge hit compared to regular ray tracing. He mentions toward the end of the paper that using distributed methods may be able to produce a better result by storing certain information and using a large number of passes. But, again, this almost sounds like photon mapping, and it would seem more appropriate to modify photon mapping to incorporate diffraction rather than partially recreate it using the approach specified in this paper. This paper made a solid step forward in making the point that diffraction can be modeled in a normal ray tracing environment. However, in terms of a realistic model, it took a step backwards with the CD wide color bands.

Though somewhat aside from the point of this research, I find it interesting that Tsingo later applied this same approach to model sound diffraction. The desire was to spatialize sound so that it was more realistic to the observer. Sound diffraction is actually much more common than diffraction of light because the wave lengths of sound are so much closer to the order of normal everyday objects. This explains why you can hear sounds from long distances, around corners, and sometimes get sound

voids and spikes as you change observing location. Tsingos approach would work very well for the every day macroscopic objects we see because most of them are comprised of right angles. For example: hallways, doors, buildings, desks, cars, ceilings, etc. Simulating these environmental objects as wedges is not really too far off from how they actually look. For this reason, I believe Tsingo's approach would do a very good job of simulating, as one example, an ambulance driving through a downtown with where you can hear the sound and where you can't hear the sound because of the interference that occurs.

The next attempt to improve diffraction simulation was the paper Rendering Iridescent Colors of Optical Disks. One of the best aspects of this paper is the excellent description of the diffraction effect towards the beginning. It is clear from their description of diffraction that the authors had a very concise understanding of the problem and how they were going to approach a new solution. Their novel approach was to treat the surface of the CD as a series of consecutive spheres that constitute the different tracks on the surface of the disk. Each of these tracks then partially superimposes itself on the surrounding tracks, giving the surface a slightly bumpy, but relatively flat, overall surface. To differentiate between the lands and grooves of the CD surface, some of these spheres were considered to be diffraction capable surfaces, and the other spheres would simply provide the spectral component of the CD surface, as the entire surface does not immediately create diffraction. Determining the guess for this spectral component was derived by looking at a vinyl disk, since these types of old records do not produce diffraction and merely spectrally reflect. This allowed them to create a guess as to how many of their spheres should reflect and how many

should diffract. They were they able to model partial periodicity of the surface for the different tracks. These items combined create a very usable CD surface simulation, especially because the individual little spheres are invisible to the observer due to their small size. The actual algorithmic diffraction used in this paper was also interesting because it relied on a modified diffraction equation using the delta function to create the interference spikes. While the logic behind their equations is sound, I do question why they seem to have avoided using any of the standard diffraction equations. That said, their results in Figure 7 look pretty good, and are close to the photographic images of optical disks.



1 (a) and (c) are photographs of a CD-ROM for different illuminating and viewing directions, and (b) and (d) are the rendered images corresponding to (a) and (c). **2** The rendered images for a point light source with different illuminating angles. **3** A rendered CD-ROM under illumination of three point light sources. **4** Interaction between a CD-ROM and a white plastic sphere.

Figure 7: The results page taken directly from Iridescent Colors of Optical Disks.

This approach was even able to capture both the major and minor strips of light on the disk surface. Mainly this is seen in that the brightest stripe shows the color

spectrum developing in one direction while the two secondary stripe how their spectrum develop the opposite direction. One of the major drawbacks to this approach is the energy conversation. As they state:

"Finally, in this paper we will focus on relative light intensity distributions because calculating the absolute values of light intensities involves more low-level physical information such as exact pit shapes, which deviates too much from our main interest – generating iridescent strips of optical disks. Thus in rendering, we will adjust the relevant intensity constants to obtain the appropriate visual effect (including proper relative intensities among the major strip, minor strips and the rest area). But in principle it is possible to determine these intensity constants from low-level physical information such as the percentage area covered by pit grooves, exact pit shapes etc."

Because of this, whenever the emitting energy exceeded their threshold, they simply clipped the value. This may or may not be adding more energy to a complicated scene, making it possibly unusable in conjunction with other algorithms. Though, in their results, they were able to render in a multiple object environment, I question how complex this environment really was. Another nice aspect of this approach is their description of the coding implementation. Using this method did not require a completely dedicated renderer to allow testing. They simply switching out the shader for the CD material surface and modeled everything from there. This allows for nice integration into almost any rendering environment which is highly desirable.

One of the most recent papers to talk about diffraction was Physically Based Real Time Diffraction Using Spherical Harmonics. The main aim of this paper was to be able to render diffraction in a real time manner. Obviously, one of the biggest hurdles to achieve this was that any of the previous approaches took a large amount of time. Their solution was to precalculate a scene's lighting and diffraction characteristics using spherical harmonics for the spectral lighting. This was accomplished by using a full hemisphere of lights in the scene and then calculating the light values around the scene. Another hurdle they had to overcome was that all current graphics pipelines for real time rely almost solely upon RGB values. This meant that they needed to use some other method to keep the entire color spectra available during their calculations which is key to modeling diffraction correctly. They chose to use the composite model because it allowed them to keep track of the spikes in the color bands caused by diffraction and also because using spherical harmonics somewhat amplifies this spiking effect. Using other methods of coloring could easily gloss over these diffraction values and lose some of the calculated data. Specifically, as they mention in the paper, "diffraction requires a more sophisticated

sampling and color representation approach than interference in order to capture these spiky peaks". While I do believe this is a very elegant approach to solving the real time diffraction problem, I do have some issues with their approach. Firstly, and most importantly, their description of diffraction in the paper is not quite well developed enough. This bothers me mainly because it implies the approach taken could have been better or have been misled due to a poor understanding of the problem space. I was somewhat surprised by this. Secondly, I have a hard time

believing that this approach would work well in a true real time environment. I do not see their approach being applied to Quake or Unreal Tournament any time soon, and if not, what real time environment are they really solving for? Doing this level of simulation seems like something more appropriately suited towards the ray tracing environment. That being said, their statement that computer graphics is going to transition over to wavelength-based phenomena system is something I do not agree with. Modern graphics pipelines are highly optimized to produce high frame rates in extremely large polygon environments. The ability to simulate certain ray-tracing-like effects, such as dynamic shadowing, has been made available through scripting enhancements such as HLSL. But, this does not imply that that a wavelength level clarity in a game environment is something anyone cares about. How many effects as this level are actually used or seen in our day to day world anyway. Many of them can be hacked around for that matter. So, with that view, if the real time graphics direction is not really going in the wavelength direction, why would using composites and spherical harmonics be something that is highly desired? I do applaud their approach for being an elegant solution to the problem space they laid out, but I do not believe that space to be completely correct, nor do I believe it could truly be used in the environment they set out for. This paper seems far ahead of its time.

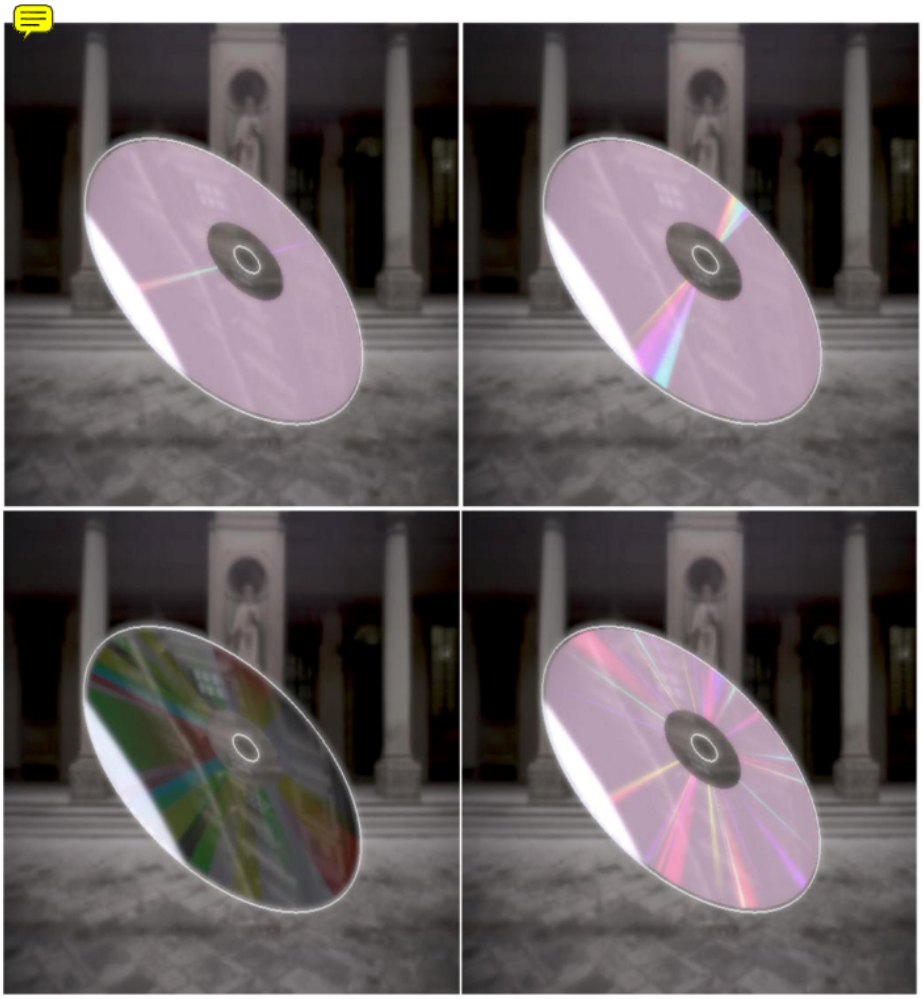


Figure 5: Our final images: Vertex shader, full effects (glow, tone mapping HDR, EM) top left: single light - 60 FPS, top right: Niave IBL 15 FPS (depending on samples), bottom left: SH - 60 FPS, bottom right: Multiple Point Lights 1-3FPS (20+ lights)

Figure 8: The results of real time diffraction.

These papers show some of the gains and shortcomings of the work done in this area. Most of them suffer from the problem of being very heavyweight in terms of calculation load. Others simply suffer from lack of realism. I believe there is much more simplistic approach to the problem.

Approach

While almost all of the previous work attempted to use very new and novel approaches to try and solve the diffraction simulation problem, I propose to go the opposite direction, and in fact, only use resources already available in a typical renderer.

I further propose that it's best to take a step back from trying to approach a CD or DVD right from the beginning. Trying to tackle the complicated situation right from the get go seems like a trap into getting focused on a specific solution. Rather than do this, I want to create a general solution that can be applied to more than one situation. Say, for example, the next Pixar movie wants to create a hero with a special suit that diffracts all incoming light. This would not be possible if using a specific optical disk based approach. To accomplish all of this, I'm going to rely upon the general ray tracing concept of a light.

In most college level physics programs, light and diffraction are taught alongside the introduction to modern physics. This is best for students because it allows them to start looking the very nature of everything as wave based at the same time.

One of the interesting experiments used to teach and show diffraction is to use a sodium based laser and a diffraction grating of known slit width and distance. Usually this grating has a slit width on the order of yellow light (relatively near the center of the visible spectrum) and a very small distance between gratings. This

means there a very few maxima, that a very far apart, and very large stretching minima areas.

What this allows for is the ability to determine, from the known diffraction equations using the grating constants, the exact wavelengths that comprise the color of sodium light. This is done by measuring the angle from the grating to the first maxima, and then back calculating what that wavelength must be to appear at that location. This allows students to calculate the spectrum that a sodium laser produces.

For the purposes of my research, this was essentially the approach I took. Instead of taking the optical disk as my base case, I essentially assume no base case, but instead a situation where you could literally repeat the sodium laser experiment.

To avoid making an overly complicated experimental situation out of this at the start, I make some assumptions. Firstly, the light incident upon the diffraction surface will essentially be like a laser. This means no stray light around the edges, and a relatively circular incident pattern. Secondly, only one dimensional vertical slits will be used for the grating to start. Having two dimensional slit patterns creates a difficult problem because the normal physics literature does not give nice equations for evaluating two dimensional slits. However, I will propose a way this can be done later in the discussion section.

Next, in order to actually make the diffraction possible using regular components, the diffraction surface will be treated like a point light source. This may seem like a strange way to approach the problem, but it actually works out rather nicely, and allows for some easy modifications to the normal ray tracing pipeline.

Specifically, any time a ray is cast into the viewing plane for a given pixel, it intersects somewhere in the scene. If this surface is just any regular old surface in the scene, its light output towards the camera is determined by iterating over the lights in the scene and determining their contribution to that surface point.

With this approach, the output from the diffraction grating is also calculated specifically for that point.

This is a very large jump from the previous work done in this area, because most of the previous works, while able to render a CD, did not appear to be able to interact with the environment they were in. This is a major shortcoming that the optical surface can only be described in terms of the incident light, but that the output light cannot be incident on the rest of the scene. In fact, it appears that more than one of the approaches would be incapable of modifications to even make this possible.

The ability to render an optical disk using this method, or any surface for that matter, will also be described in the discussion section because it goes beyond the core idea of this approach.

It can appear odd that this approach is targeted initially towards using a point light. However, because the diffraction surface is so small relative to the size of scene, it is somewhat irrelevant that the diffraction grating has any size at all. A lot of the dimensional components are actually handled by the algorithm itself, which will now be discussed.

Algorithm

The biggest consideration for rendering diffraction was the ability to accurately represent the slits in the grating. Numerical approaches could be taken to determine if the wavelength and the aperture are of appropriate size to cause diffraction, but this quickly runs into the over complications of some of the previous work.

After some thought, the process could be simplified down to treating the actual diffraction grating as a light source. This immediately removes a great deal of the complication and allows the diffraction to be placed directly within the typical rendering coding solution. This approach is also nice because it allows for easy future expansion in the more complex direction.

The first step in treating the diffraction grating as its own light source is to define the constants needed for the grating.

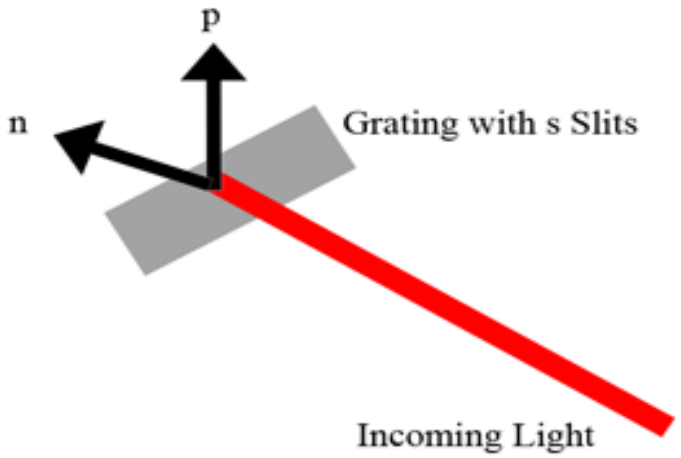


Figure 9: Defining constants for the diffraction grating.

The main constants involved are the normal to the grating plane *n*, and the grating direction defined in the direction of the slits *p*. This can be seen in **Figure 9** where the grating is defined by *n* and *p*. The other parameters needed for the light source are the number of grating slits *s* and the diameter of the incoming light over these slits. The diameter becomes part of a special hack later, but is actually not relevant to the diffraction equations.

For the purposes of rendering the diffraction pattern in the paper, this approach also helps because the grating and the incident light can be completely ignored. They can simple be assumed. The discussion section will talk more about using lights in the scene to specifically dynamically adjust this value.

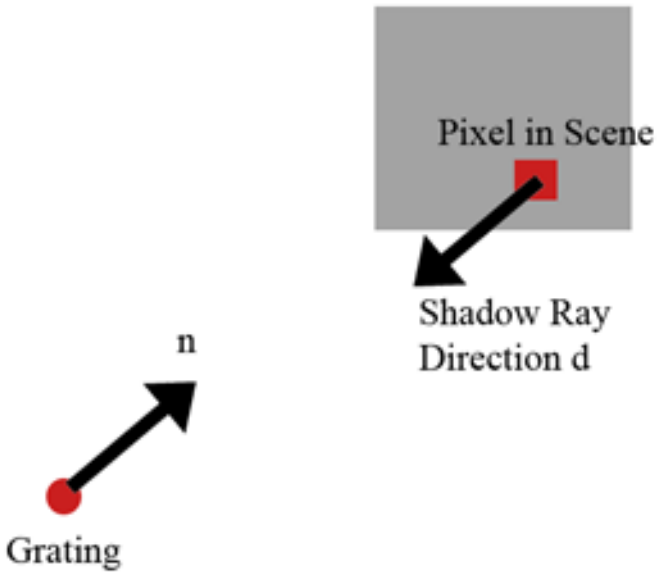


Figure 10: Runtime calculation begins to render the diffraction pattern.

Because this is all that is necessary to define the grating, the next step is to actually render the scene. As with a traditional rendering environment, each pixel in the output picture is traced into the environment. Wherever this ray hits then runs through all the possible light sources and determines each of their individual contributions as in **Figure 10**.

One of the things that makes this a little easier is that a diffraction grating only shows a pattern on the outgoing side of the grating, so any time the dot product of the shadow ray direction and the grating normal *n* is negative, we can safely assume the grating adds no contribution.

Trying to figure out where in this hemisphere the grating lies is somewhat more complicated, however.

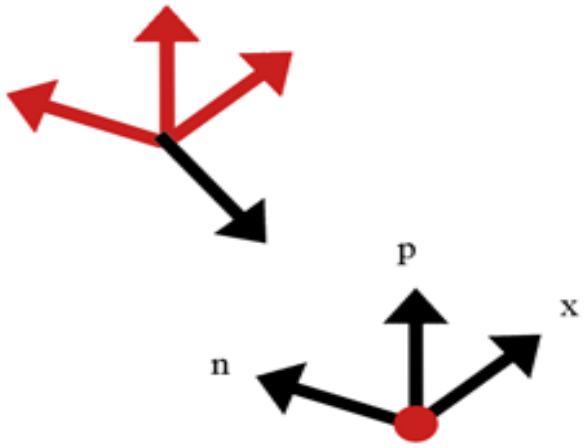


Figure 11: Breaking down the hemisphere into useful components.

Firstly, the shadow ray direction needs to be broken down into what are essentially the origin components of the grating constants. Firstly this is done by calculating the missing orthogonal parameter that defines the grating origin.

As in **Figure 11**, this is defined as the cross product of grating values *n* and *p* which I will call *x*. The shadow ray direction dot product is then taken against *n*, *p* and *x*. This allows the individual components that define the shadow ray relative to the diffraction grate to be combined into new plane dependant vectors.

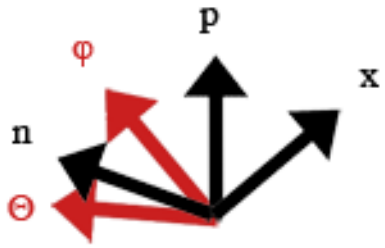


Figure 12: The plan dependant vectors.

This can be seen in **Figure 12**. The previously calculated dot products are now used to recreate θ and φ. θ is the vector defined by the shadow ray direction mapped into the plane defined by *n* and *x*. φ is the vector defined by the shadow ray direction mapped into the plane defined by *p* and *n*.

θ is the single most important part of this process because it is the main parameter dictating the diffraction equations and the pattern they produce. Similarly to the diameter of the incoming light beam to the grating, φ is going to be used for a form of hack to create realism in the resulting diffraction pattern.

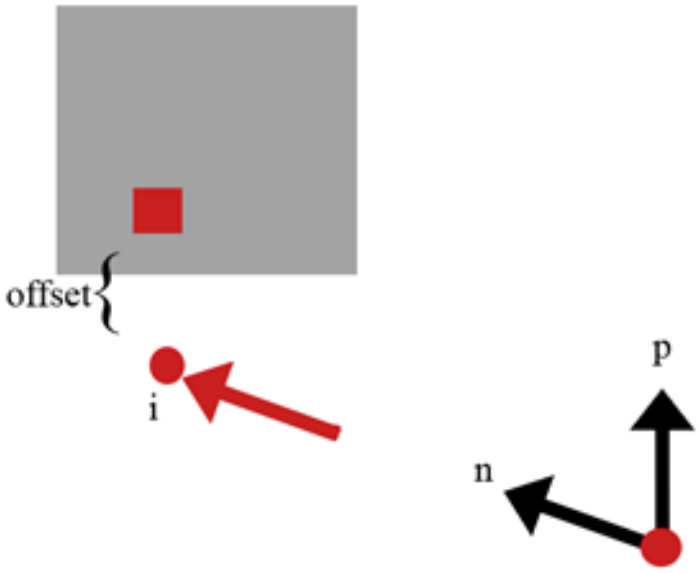


Figure 13: Finally narrowing down if the pixel traced into the scene is contained with the diffraction band created by the grating.

At this time, all the necessary variables are available to finally make the calculations as to the contribution of the diffraction grating to the given pixel in the final image.

First, the overall vector describing the difference in position between the diffraction grating and the scene intersection point is use in a dot product with the grating normal. This gives the distance to the plane defined by the scene intersection point and the grating normal from the grating. This distance is then used to calculate the point in front of the normal that lies in the previously described scene position plane that I will call *t*.

The distance from *t* to the scene position is then used in a dot product with the normal defining the slit direction *p*. This value, as seen in **Figure 13**, gives the distance away from the central plane of the diffraction grating definition. This allows the diffraction pattern to have a width that is determined by the value of the incoming beam.

At this point, θ can be used in the traditional diffraction equations to determine the output pattern.

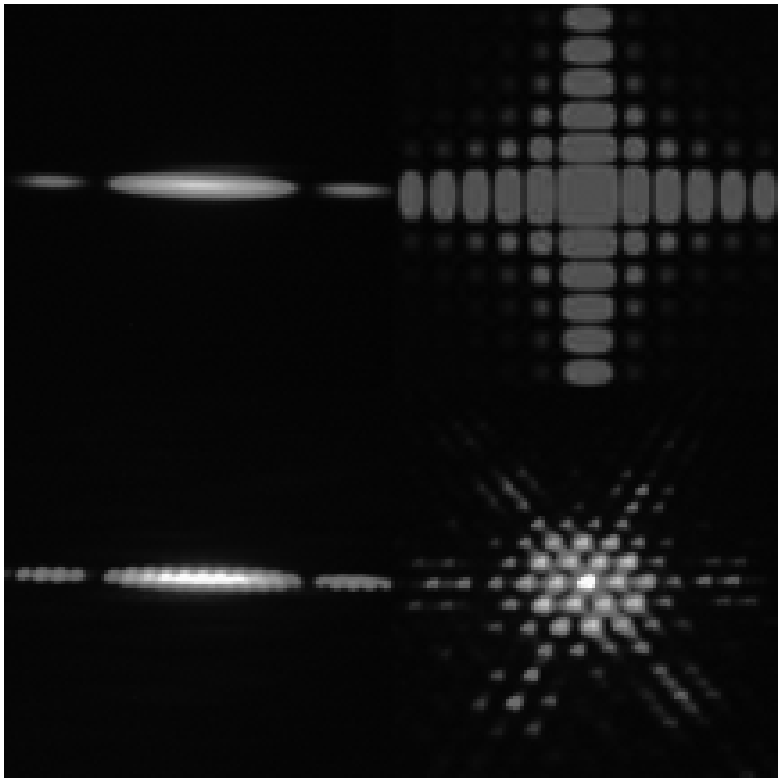


Figure 14: A repeat of Figure 3, showing real world diffraction patterns [15]

One thing that is not accounted for in any physics textbook I could get a hold of is the way real diffraction patterns actually appear, as shown in Figure 14. The

equations and algorithm described so far tends to appeared very squared off at the edges of the diffraction regardless of parameters. This is because the problem is essentially only defined for two dimensional space. The three dimensional solutions almost instantly become too complicated for use with this approach. However, the two dimensional equations produce a highly accurate result that looks nearly identical to reality. A sample of output from the algorithm, which appears very square, is shown in Figure 15 and accurately represents the standard physics equations. What was causing this elliptical edging?

None of the advanced physics reference books or journals I searched through included any reasoning or explanation for why the diffraction patterns always appears to be elliptical in reality rather than square. The equations were obviously correct, but the real output was doing something extra that took quite a while to indentify.

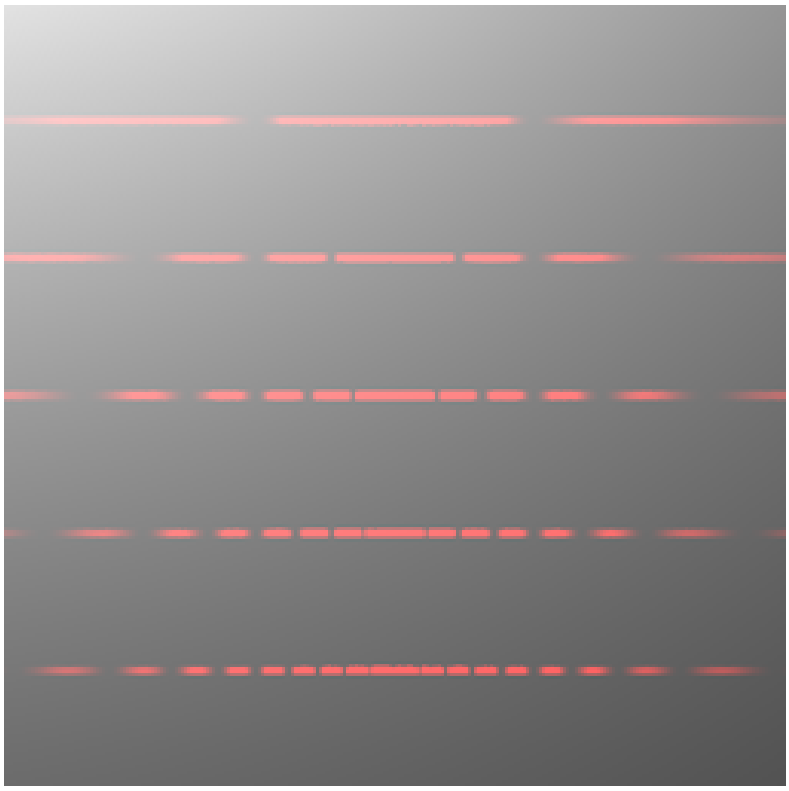


Figure 15: Output from the algorithm based off the straight physics description.

What I realized, was that the diffraction pattern was attenuating for decreased intensity based of the incident light assumption. This means that the farther up or down from the center of the incident light beam, the less intensity was being incident on the grating, and therefore attenuating the pattern in a three dimensional way.

To account for this, I assume that the incoming light to the grating is circular. This allows intensity at the distance offset from the 1-D diffraction grating calculation to be attenuated by the amount of width present at that same offset up or down on a circle. This attenuated intensity can then be further attenuated by using φ if desired, but for the purposes of my results images, this will not be done.

This method works well for any sort of one dimensional vertical slit grating.

Code

This code is added to the shader function for any rendering material and occurs for each light source. Any regular light would fall into the 'else' of the initial 'if' statement.

```
if( pLight->dif ) // determine if the light is a diffraction grating
{
	// if on the outside of the grating
	if( dot(shadowray.d,pLight->m_normal) < 0 )
	{
		// determine the contributing dot values of the ray in the direction of
		// the hit
		Vector3 x = cross(pLight->m_normal,pLight->m_pattern);
		float a = abs(dot(shadowray.d,x));
		float b = abs(dot(shadowray.d,pLight->m_normal));
		float c = abs(dot(shadowray.d,pLight->m_pattern));

		// map the shadow ray onto the normal/pattern plane
		Vector3 phi = b*pLight->m_normal+c*pLight->m_pattern;
		phi.normalize();

		// map the shadow ray for the theta for the diffraction calculation
		Vector3 theta = a*x+b*pLight->m_normal;
		theta.normalize();

		// determine where this line intersection with the view plane for
		// attenuating the pattern
		Vector3 t1 = hit.P-pLight->m_position;
		float diffractiont = dot(t1,pLight->m_normal);
		Vector3 planePoint = pLight->m_position + pLight->m_normal*diffractiont;
		t1 = planePoint-hit.P;
		float splinedistance = abs(dot(t1,pLight->m_pattern));

		// narrowing the band
		if( splinedistance <= pLight->diameter/2.0 )
		{
			float sintheta = dot(theta,pLight->m_normal)/(theta.length()*pLight->
				m_normal.length());
			sintheta = sqrt(1-sintheta*sintheta);

			for(int color=0; color<3;color++)
			{
				float a = 0.000001;	// hard coded slit width in meters
				float b = 0.000002;	// hard coded slit spread in meters
```

```
                                        float lambda;
                                        if( color==0 )
                                                lambda = .000000450;    // red light
                                        else if (color==1)
                                                lambda = .000000550;    // green light
                                        else
                                                lambda = .000000700;    // blue light

                                        float delta = 2*PI*sintheta*a/lambda;
                                        float intensity = sin(delta/2);
                                        intensity = (intensity*intensity)/((delta/2)*(delta/2));

                                        if(pLight->slits>1)
                                        {
                                                float xi = 2*PI*sintheta*b/lambda;
                                                float multiintensity = sin(xi/2*pLight->slits);
                                                multiintensity =
                                                        (multiintensity*multiintensity)/(sin(xi/2)*sin(xi/2
                                                        ));
                                                intensity *= multiintensity;
                                        }

                                         // the integral this creates makes for interesting discussion.
                                         // for now, clamp it
                                        if( intensity > 1.0 )
                                        {
                                                intensity = 1.0;
                                        }

                                        float circlex = sqrt(1-pow(splinedistance/(pLight->
                                                diameter/2.0),2));
                                        float ratio = pow(circlex,2) ;//* pow(dot(phi,pLight->m_normal),4);

                                        // some previous attempts that look slightly different
                                        //float circlex = sqrt(1-pow(splinedistance/(pLight->
                                        //              diameter/2.0),2.0));
                                        //float ratio = ((circlex)*2.0)/pLight->diameter;
                                        //float ratio = 1.0-splinedistance/(pLight->diameter/2.0);

                                        Vector3 colorvec;
                                        if( color==0 )
                                                colorvec = Vector3(1.f,0.f,0.f);
                                        else if (color==1)
                                                colorvec = Vector3(0.f,1.f,0.f);
                                        else
                                                colorvec = Vector3(0.f,0.f,1.f);

                                        L += ratio*intensity*pLight->color()*pLight->wattage()*colorvec;
                                }
                        }
                }
        }
```

Similarly, the next code segment is for the CD. The algorithm is essential identical. Many values are hardcoded here for example purposes.

```
Vector3 Point(4.0, 3.5, -3.0);
Vector3 Normal(-2.0, 1.0, 1.0);//(-1.0, 10.0, 5.0);
Normal.normalize();

Vector3 axis = hit.P-Point;
Vector3 patterN = cross(axis,Normal);
patterN.normalize();

Vector3 x = axis.normalize();
float a = abs(dot(ray.d,x));
float b = abs(dot(ray.d,Normal));
float c = abs(dot(ray.d,patterN));

// map the shadow ray for the theta for the diffraction calculation
Vector3 theta = a*x+b*Normal;
theta.normalize();

float sintheta = dot(theta,Normal)/(theta.length()*Normal.length());
sintheta = sqrt(1-sintheta*sintheta);

for(int color=0; color<3;color++)
{
	int slits = 4;
	float a = 0.000006;
	//////////123456789
	float b = 0.0000083;

	float lambda;
	if( color==0 )
		lambda = .000000450;
	else if (color==1)
		lambda = .000000550;
	else
		lambda = .000000700;

	float delta = 2*PI*sintheta*a/lambda;
	float intensity = sin(delta/2);
	intensity = (intensity*intensity)/((delta/2)*(delta/2));

	if(slits>1)
	{
		float xi = 2*PI*sintheta*b/lambda;
		float multiintensity = sin(xi/2*slits);
		multiintensity = (multiintensity*multiintensity)/(sin(xi/2)*sin(xi/2));
		intensity *= multiintensity;
	}
```

```
        if( intensity > 1.0 )
        {
                intensity = 1.0;
        }

        Vector3 colorvec;
        if( color==0 )
                colorvec = Vector3(1.f,0.f,0.f);
        else if (color==1)
                colorvec = Vector3(0.f,1.f,0.f);
        else
                colorvec = Vector3(0.f,0.f,1.f);

         L += intensity*colorvec*pLight->color();
}
```

Results

First, note that because of the values involved for ξ and δ, the slit width and distance can make a very pronounced effect on what the resulting diffraction pattern appears like.

To run tests, a simple Cornell box was used with some diffraction gratings present. These results were taken by looking at the wall in the direction of the diffracted light.

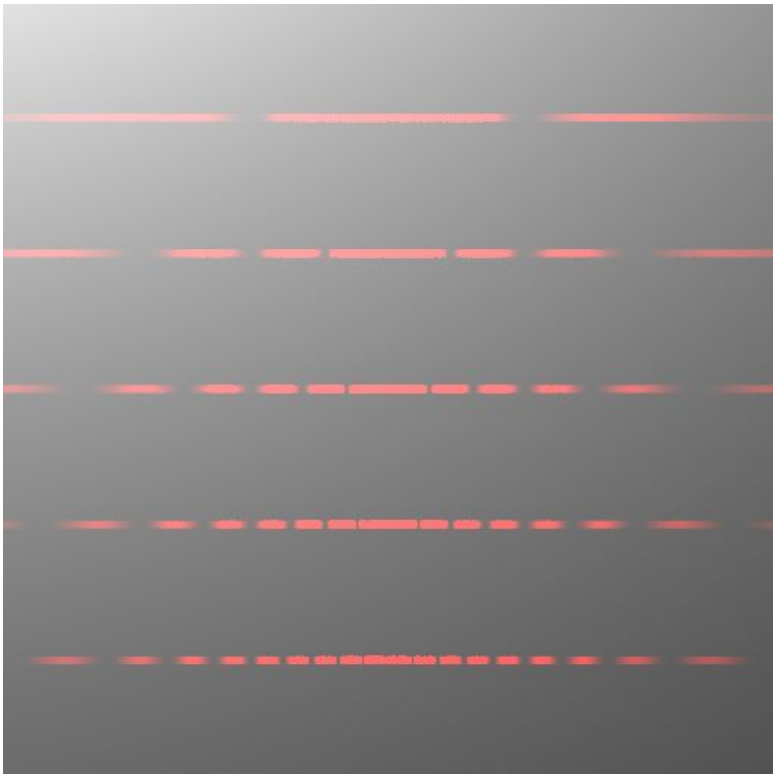


Figure 16: The original multiple diffraction gratings where the results appeared very squared off. The top is a single slit, followed by 2, 3, 4, and 5 slits.

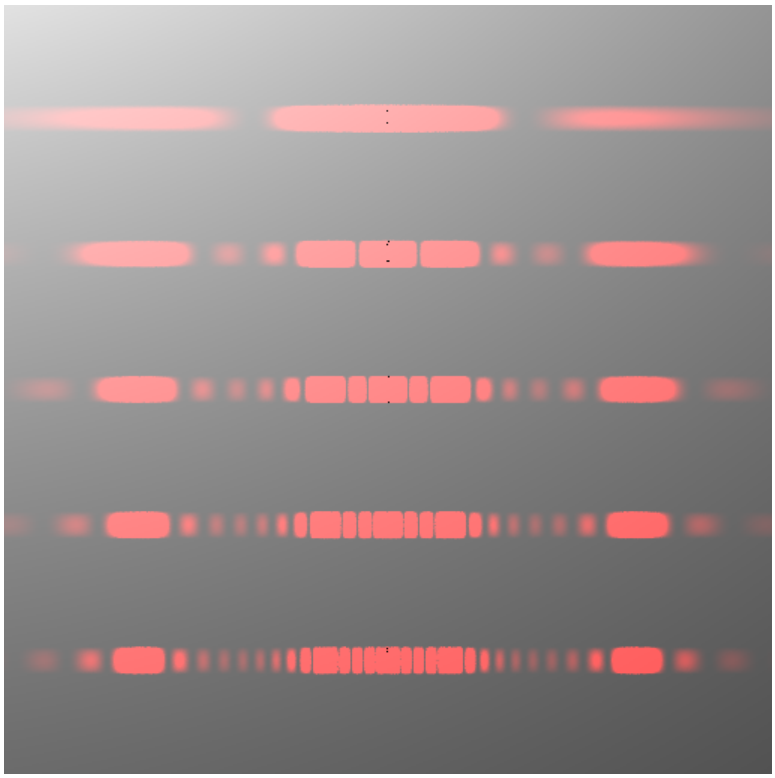


Figure 17: The attenuated multiple diffraction gratings where the results now appear to be elliptical. The top is a single slit, followed by 2, 3, 4, and 5 slits.

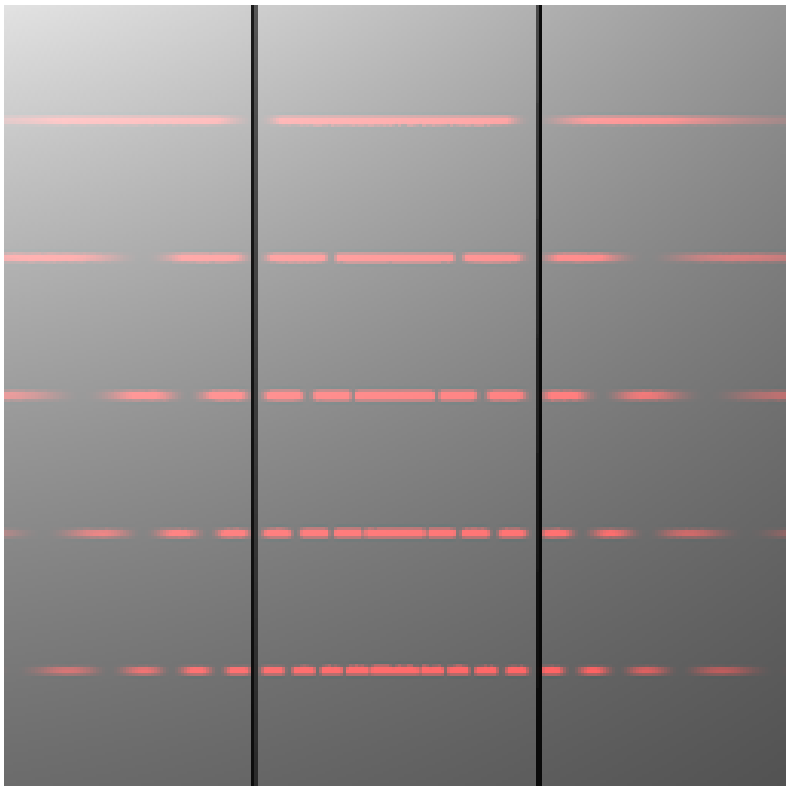


Figure 18: (From un-attenuated output) Here the minima from a single slit are highlighted so that the multiple slit minima and maxima can be seen to align to the same points. It can also be seen the intensity issue that arises from having multipliers greater than 1.0. Even though the multiple slits intensity should always be less than the single slit intensity for a given point, this isn't quite always the case. The 5 slit pattern is still quite bright right up to the minima edge, whereas the single slit is quite faded out in this region.

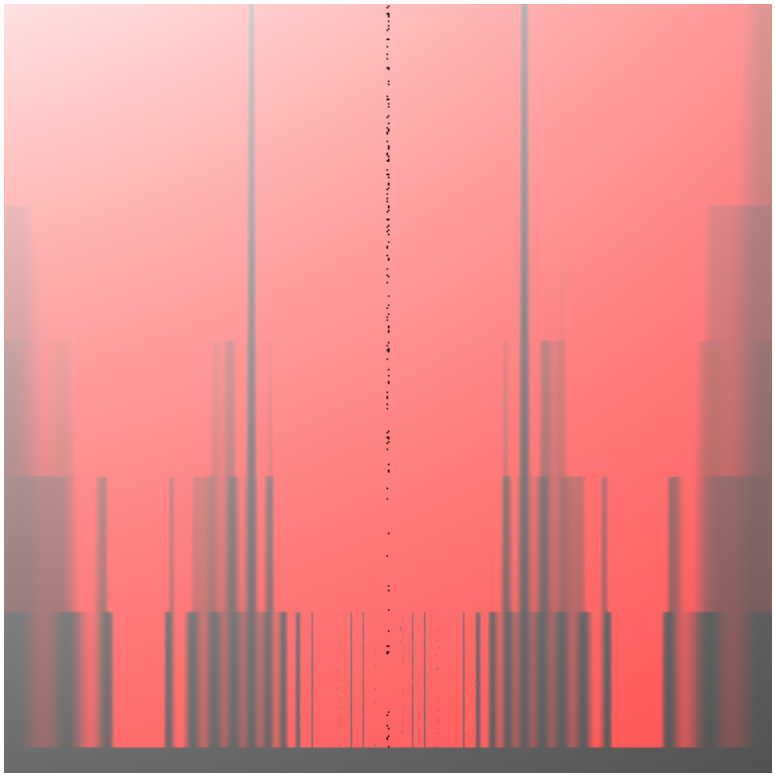


Figure 19: Similarly, this was an erroneous output but actually displayed the minima lining up very well. Each increasing set of minima lines up with the minima of a lesser order slit value.

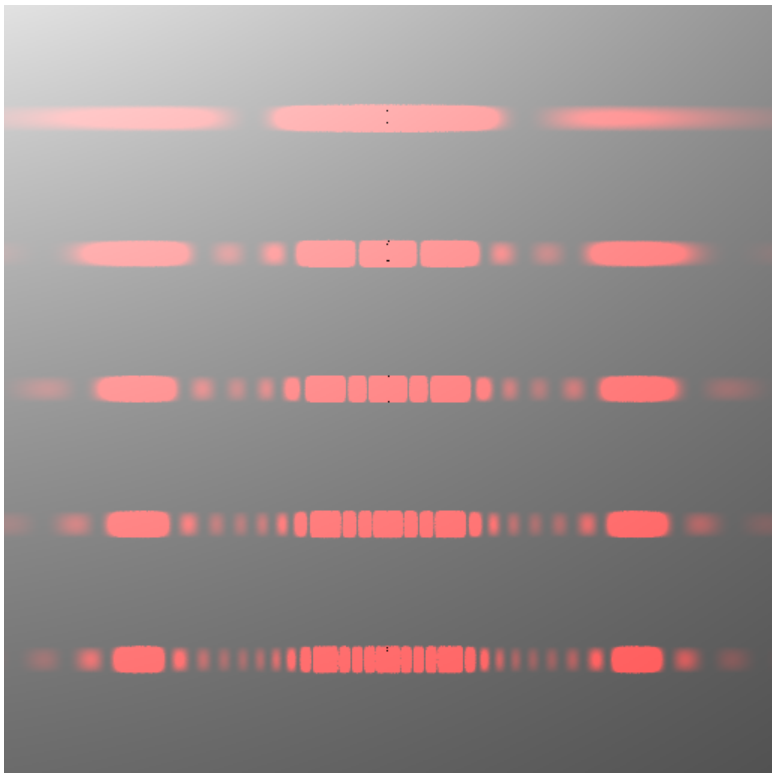


Figure 20: Red value.

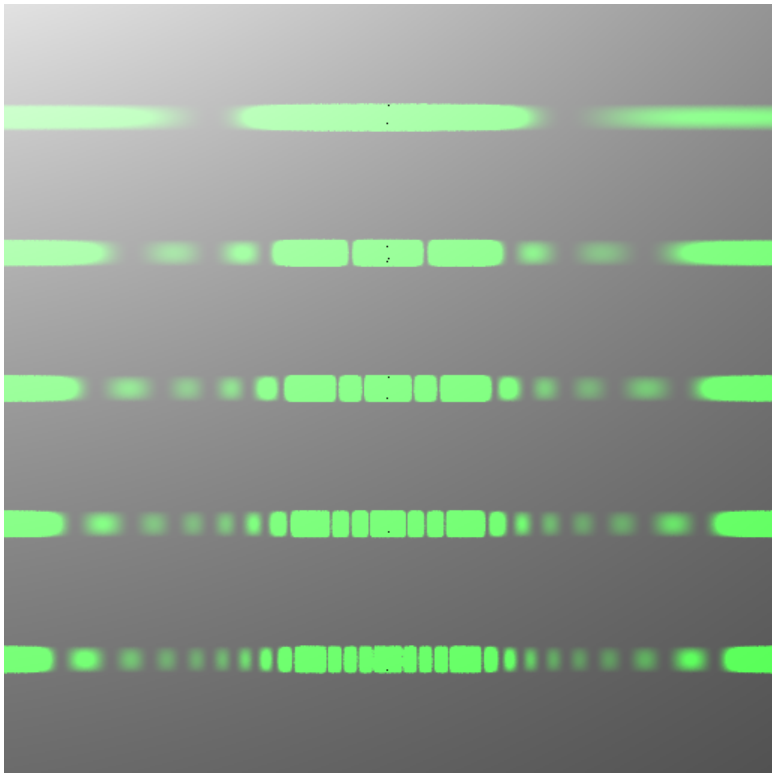


Figure 21: Green Value

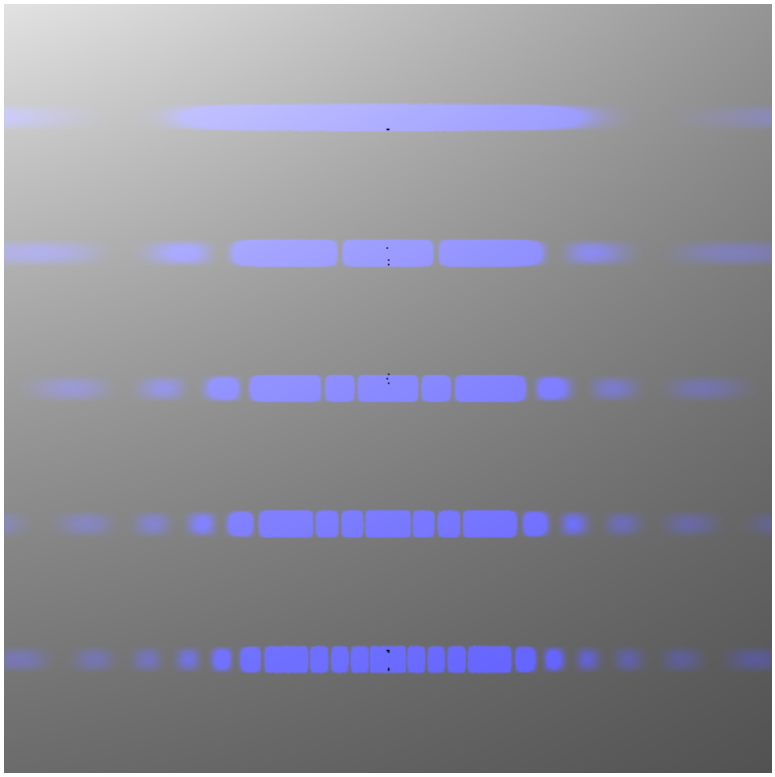


Figure 22: Blue value.

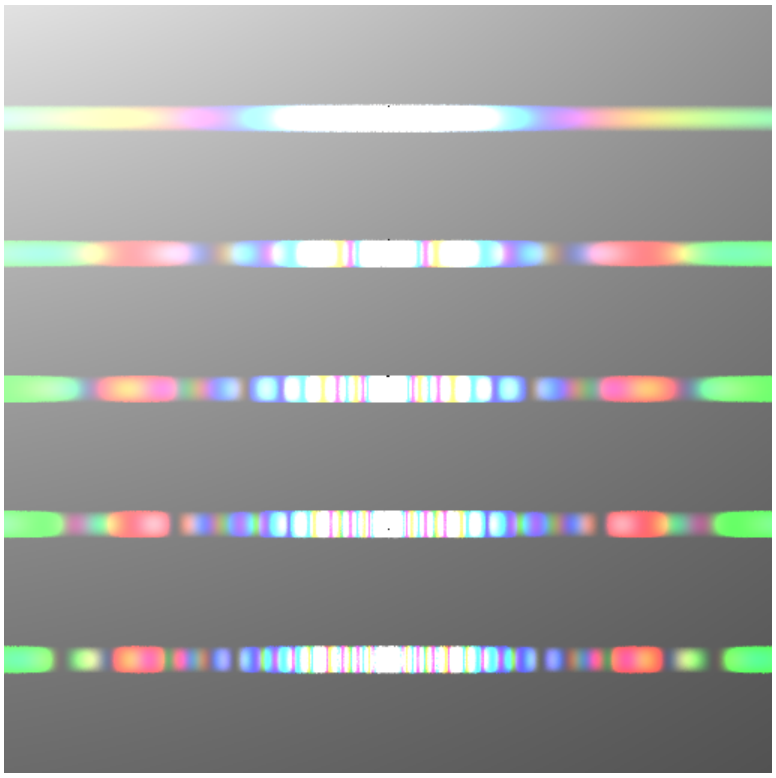


Figure 23: The RGB values combined.

The implemented algorithm produced some good images that demonstrate multiple slit diffraction. In Figure 16, the results are displayed form a straight on view. There are a few artifacts along the edges of the diffraction, but these are no related to the diffraction algorithm and instead come from the viewing angle of the scene. Figure 17 shows the same diffractions with the attenuation of the incident light source shown. Figuring this out was one of the most difficult parts of this research.

One good indicator that the simulation went well is evident in Figure 18, where the minima are the same for all the different diffractions. The maxima for all the gratings also fall within the maxima of the single slit as well. Looking at Figure 19

shows this even better by showing that the minima consistently go through all the slit orders from one on up.

Looking at Figure 20, Figure 21, and Figure 22 shows exactly what we would expect with different wavelengths. As the wavelength increases towards the blue end of the spectrum, the diffraction pattern begins to widen. This is where the prism-like effect that one sees on an optical disk comes from. To show this more clearly, Figure 23 uses all three color channels as independent calculations and accumulates their results. This would be where using spectral light values would achieve more realistic results. But, for the purposes of this paper, this adequately demonstrates the algorithm.

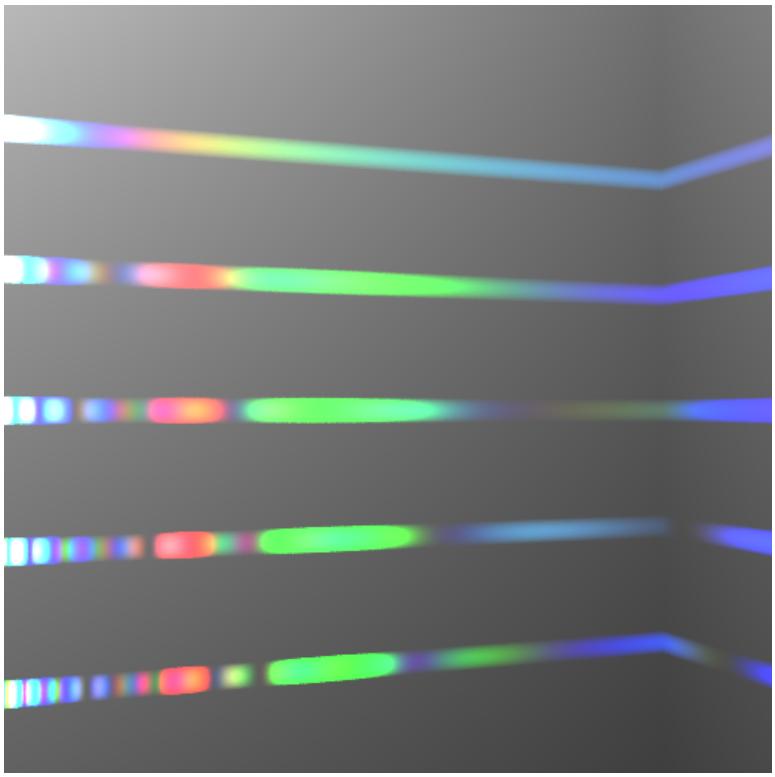


Figure 24: The diffraction pattern dynamically rendering onto the scene.

The other big thing to note in the results is that the algorithm renders itself onto the scene by its intersection point as a dynamic calculation. This can be seen in Figure 24 where the side wall is calculated on the fly differently that the back wall where the grating is pointed.

For comparison sake, I also tried a test rendering of a CD with some interesting results. In order to render the CD, the algorithm required almost no changes, though a few assumptions could be make that deviated from the code listed previously. Firstly, you can always assume that the CD surface is diffracting, secondly, there is no band narrowing, and thirdly, I made the assumption that any point you view on the surface of the CD is comprised of a small diffraction grating.

Rather than treating the whole CD as one large diffraction grating, treating the intersection point on the CD surface as one individual diffraction grating origin greatly simplifies the overall calculation. This may not necessarily be the optimal approach, but it avoids the huge pitfall of some previous work that not viewing each calculation point as a smaller section of the whole CD then requires an approximation of how the entire CD surface works together at one time. This occurs even though the viewing area, or one pixel in the end, is only a very small part of the surface. I believe this to be a good and safe assumption on my behalf.

One thing that prevented a direct comparison with previous work was the lack of parameters listed in any of the papers. To do a side by side comparison would require knowing the parameters the previous papers used. For the purposes of my work: each picture was 600x600, the CD surface was assumed to have a 12 centimeter

diameter with a 2.5 centimeter inner hole, 0.6 micrometer groves, 0.83 micrometer lands, 4 grating slits per intersection point, a 60% reflection at all spectrums, red light was 450 nanometers, green light was 550 nanometers, and blue light was 700 nanometers.

One to note on these pictures is that there is a string of artifacts along the main maxima line. I have not yet isolated the source of this.

Figure 25: A test rendering of a CD using the same algorithm with a light to the side.

Figure 26: A test rendering of a CD using the same algorithm with a light to the top and relatively far away.

Figure 27: A test rendering of a CD using the same algorithm with a light to the top and relatively close to the CD.

In Figure 25, Figure 26, and Figure 27 you can see the banding of colors that occurs along the grooves of the CD. These figures appear relatively realistic, but suffer from some slightly odd banding and not quite as much color shift on the surface as I had hoped for. However, that being said, changing the parameters only slightly creates a noticeable difference in the CD appearance. Perhaps my parameters are only slightly off from the true physical model or my slits assumption is glazing over some aspect of the CD, but this seems very close.

After these samples were made of the CD, I tried for a very long time to go back and improve upon their appearance. I tried changing some of the math to situations that were tweaked from realistic and situations that were completely

unrealistic. I tried altering the size and surface parameters of the CD. I tried moving the light source around. I tried looking for the banding problem. Almost everything I did ended up making the CD appear less realistic. The one exception to this was keeping the algorithm as described in this paper and modifying the parameters used for the diffraction. That had the greatest effect of all. However, the effect for even the smallest change was so profound that I am left wondering if my somewhat realistic results are falling outside of a very slim band of parameters that produces photorealistic results.

I can only hypothesize that this may be the case, because every after a few hundred attempts, the best results have already been shown. Due to the fact my approach was significantly different than all previous attempts, I am left somewhat stuck and without a direction to move forward with. My approach may very well be correct, but at least for its application directly onto a CD surface, I am not seeing the tweak necessary for photorealistic results.

However, compared to previous results, I am looking slightly better than most of the examples from previous work. The main contribution that I can offer is that my approach not only contributes to the scene itself, but also relies on the basic physics principles of diffraction and does not use anything that wouldn't already be in a normal rendering program anyway.

Discussion

The results show very realistic simulations of diffraction in a practical context. Because of the simplicity of using existing elements to create the simulation, the method can be extended to simulate a number of more complicated situations without too much hassle. For its realism, however, there are certain drawbacks and certain unanswered questions regarding this method, though. All of the extensions, drawbacks, and questions will be covered.

One of the most significant items of note is that this method contributes to the overall rendering environment. This was a major drawback of the previous work done in this area. This is easily seen in that the results pictures show the output of the diffraction on the wall of the scene. Because of this calculation method, any number of lights and gratings can be present in the scene and contribute to each other.

To do so would only require that for every grating that can contribute to a surface, that grating iterates over all other lights in the scene and accumulates their contributions, whether they be lights or gratings themselves, into the scene point contribution. Any cross contribution between gratings is easily handled in the same way that a ray bouncing between two mirrors stops recursing after a certain number of reflections. The only downside to this is that the output of the original grating is not factored into the contributions of the other gratings, though this could be handling by doing a single round of precalculated contributions from any grating to all other gratings.

Another big item to mention is the color of the diffraction. In order for the color to be correctly represented, a spectral trace would need to occur. This is because the diffraction pattern is highly wavelength dependant. One final scene points have been computed, they can easily be converted back to typical RGB values. This would allow for the full simulation of diffraction to be correctly shown.

Of special note, and specifically the thing that started this research, is how one would achieve simulating an optical disk. This is actually not a big step from the work already done. The key to doing this is to treat, for example, a CD surface as a dynamic point calculated surface. What this means is that any intersected point on the surface of the CD is calculated as being the given CD's diffraction parameters with the grating direction using the radial angle of the intersection point with the CD. This allows to entire CD to be calculated without having to define any sort of special system for just the CD. It's just like a traditional volumetric light.

The other interesting benefit of this is that the scene can now use the CD in the same way. Using the same methods that a point light would use, the CD would be sample many times, and the appropriate pattern would appear on the surfaces of the scene.

Note though, that with using this, it must still be factored in that the CD surface still reflects a large significant portion of the incoming light in a spectral manner.

The intensities using this method are a source of concern for me, and possibly one of the drawbacks. The intensity equation, while providing the entire ability to simulate diffraction, gives intensity multipliers that are at times very large. This

means that the intensity of the resulting point of light can actually be over a factor of 10 brighter than the original light source.

To determine the correct weight necessary to properly light the diffraction emission, the following integral would need to be solved across the spectrum, and across the emission area. But, the question is whether of not this weight should be constant among all points or changing due to the emission angle.

$$\oint_A \int I_0 \frac{\sin^2\left(\frac{\pi a \sin\theta}{\lambda}\right)}{\left(\frac{\pi a \sin\theta}{\lambda}\right)^2} \partial\lambda \partial A = 1?$$

Is this equation correct? I'm not sure.

How this would integrate over a banded hemisphere becomes a confusing equation. It also becomes more convoluted by there not being a distinct equation to describe why the non-primary maxima are not as wide as the primary maxima.

Most of the problem here is caused by the fact that none of the physics textbooks I could get a hold of gave any indication as to how to handle real incident light onto the grating, which never has a two dimensional shape. Because of this, extra shapes and ellipses are caused in the diffraction output which then makes attempting the overall intensity integral a no nonsense problem.

It is possible that implementing a two dimensional diffracting grating could get closer to solving this problem. However, since the incident light is rarely shaped in

that same squarish manner that most of the gratings are shape, the problem would likely resurface in a different way. This would not be a big issue, except that this method could theoretically be extended to photon mapping.

For the purpose of this paper, a clamp was used that simply checked if the intensity value was greater that 1.0 and brought it back down to 1.0 as the maximum allowed intensity. This is not completely seen in the output however, because RGB can only support a certain spectrum and intensity of light. But, with a brighter surface, the center areas of the resulting maxima would appear almost burned out.

One performance note is that I used math library sines in order to calculate the desired interference and intensity levels. While this is great because the interference is relatively constant along a given angle, this is horribly inefficient from a performance perspective. If a given scene had a lot of diffraction information present, this would become a bottleneck very quickly.

It is possible to avoid using these sines, but it comes at a cost that most physicists don't even mention in these equations. In order to take the sine out, the offset from center over the distance is used as an approximation. This seems fine at cursory glance, but it is really using a small angle approximation. This means that only a narrow gap in front of the diffraction surface normal is really accurate to the interference present. It also means that if the receiving surface it complicated, meaning multiple surfaces, or if the surface is at an angle, each point is going to have to recalculate the distance and offset from center to use this approximation. One cannot simply use the hypotenuse because that hypotenuse is relative to the emitting phase angle. Only the sides of this triangle are constant.

This is good for efficiency purposes and would be something that would go by unnoticed, but to properly simulate the diffraction effect, sines must be used. This is obviously a very heavy operation, but there is no way around it for any non-trivially small angle from the diffraction normal. This does create a concern for any scene in which a significant portion of the scene is comprised of diffraction. Especially if multiple spectra are being used, since each individual spectra is going to need its own sine calculations. If the scene is being rendered with a real visible spectrum, this could get very computationally intensive, very quickly.

Future work continuing from this point would include the following:

- Figuring out the intensity weighting! As discussed, this may or may not be a simple calculation. By my guess, the value will not be constant and will depend on the angle of the interaction. This value will also be especially sensitive to the incident light shape.
- Creating circular slits. This could potentially be a small modification to the existing algorithm as the circular slit is the simplest two dimensional slit since it is shaped the same in all directions.
- Implementing two dimensional slits. This may or may not have a general equation to help describe. Likely, the only way to simulate this properly would be through photon mapping since the geometry of the slit would need to be taken into account to properly simulate the effect and this may be too

complicated for a back traced calculation. In fact, I will suggest that this will be too heavy for normal ray traced calculation.

- Photon mapping the diffraction grating. This would be tricky because very precise measurements would need to be make of where the photon is in the slit. Phase angles would then be combined on relevant surfaces. These could also be precomputed by combining local phase interactions and summing them into one point of overall interference which would then be used at runtime to determine contributing value. Essentially, this would be an extension to the caustic mapping.

Conclusion

The results came out pretty nicely. One thing that I did not show is that the technique used to implement this simulation allows for any distance, any number of receiving surfaces, and any angle of the diffraction. This is partially seen by the nature of the dynamic calculations against the surfaces of the scene. This flexible nature is crucial for this to be a viable algorithm.

Compared to previous work in the area, this algorithm provides some much needed extensions and also takes a necessary step back to create a generalized diffraction which could then be applied to just about any surface or situation. It is also much different than previous work in that it is not computationally intensive, but instead, relies upon tried and true standard physics equations to model the diffraction output. This is what allowed the standard rendering environment to be usable in creating this algorithm. It also means there is no question as to the accuracy of the model.

I'm happy with the progress made so far, but there is still a lot of work to be done. One of the major things is to extend this algorithm to photon mapping. However, for this to be possible, the overall intensity equation needs to be solved in a way that conserves energy. At the moment, it is difficult to tell exactly what is happening with the energy, though it clear that energy is be added and clamped. Once photon mapping and has been figured, this will have become a viable algorithm to

include in any renderer. At the moment, the rendering of diffraction will have to remain in the on the fly portion of the calculations.

Bibliography

[1] TSINGOS, N., FUNKHOUSER, T., NGAN, A., and CARLBOM, I. 2001. Modeling Acoustics in Virtual Environments Using the Uniform Theory of Diffraction. In *Proceedings of ACM SIGGRAPH 2001*, ACM Press / ACM SIGGRAPH, New York. E. Fiume, Ed., Computer Graphics Proceedings, Annual Conference Series, ACM.

[2] STAM, J. 1999. Diffraction Shaders. In *Proceedings of ACM SIGGRAPH 1999*, ACM Press / ACM SIGGRAPH, New York. E. Fiume, Ed., Computer Graphics Proceedings, Annual Conference Series, ACM.

[3] TSINGOS, N. 2000. A Geometrical Approach to Modeling Reflectance Functions of Diffracting Surfaces. Proceedings of the Eurographic's Workshop on Rendering 2000.

[4] SUN, Y., FRACCHIA, D., DREW, and CALVERT, T. 2000. Rendering Iridescent Colors of Optical Disks. Proceedings of the Eurographics Workshop on Rendering Techniques 2000.

[5] LINDSAY, C. and AGU, E. 2006. Physically-Based Real-Time Diffraction Using Spherical Harmonics. International Symposium on Visual Computing 2006.

[6] SERWAY, R. and BEICHNER, R. 2000. Physics for Scientists and Engineers with Modern Physics. Fifth Edition, Saunders College Publishing.

[7] SHIRLEY, P., ASHIKHMIN, M., GLEICHER, M., MARSCHNER, S., REINHARD, E., SUNG, E., THOMPSON, W., AND WILLEMSEN, P. 2005. Fundamentals of Computer Graphics. Second Edition, A K Peters.

[8] PEDROTTI, F. and PEDROTTI, L. 1993. Introduction to Optics. Second Edition, Prentice-Hall Inc.

[9] BORN, M. and WOLF, E. 1999. Principles of Optics. Seventh Edition, Reissued by Cambridge University Press.

[10] KINGSLAKE, R. 1965. Applied Optics and Optical Engineering. Academic Press Inc.

[11] JENSEN, H. 1996. The Photon Map in Global Illumination. PhD Dissertation.

[12] KELLER, J. 1962. Geometrical theory of diffraction. J. of the Optical Society of America, 52(2):116-130.

[13] Diffraction Article from Wikipedia.com (en.wikipedia.org/wiki/Diffraction). Accessed November 1, 2008.

[14] Multislit Diffraction Article from Hyper Physics website (hyperphysics.phy-astr.gsu.edu/hbase/phyopt/mulslid.html). Accessed November 1, 2008.

[15] Light as a Wave article from website (www1.union.edu/newmanj/lasers/Light%20as%20a%20Wave/light_as_a_wave.htm). Accessed November 1, 2008.