Ray Tracing Dielectrics with Spatially Varying Properties

By
Ahmad Ali Elhoni

Supervisor
Dr. Mohammed M. Elammari

Co-Supervisor
Dr. Kevin G. Suffern

This Thesis was submitted in Partial Fulfillment of the Requirements for Master's Degree of Science in Computer Science.

University of Benghazi
Faculty of Information Technology

June 2019

University of Benghazi **Faculty of Information Technology**

Department of Computer Science

Ray Tracing Dielectrics with Spatially Varying Properties

By
Ahmed Ali Elhoni

This Thesis was Successfully Defended and Approved on **15.6.2019**

Supervisor: Dr. Mohammed M. Elammari

Signature**:** **……M. Elammari……..**

Co-Supervisor: Dr. Kevin G. Suffern

Signature**:** **……Kevin Suffern……..**

Dr……..……..……. **(Internal examiner)**

Signature: ……..

Dr……..…… **(External examiner)**

Signature: ……..…….

(Dean of Faculty) **(Director of Graduate studies and training)**

In The Name of Allah, The Most Beneficent, The Most Merciful

" He has taught you that which you did not know, and ever has the favor of Allah upon you been great "

Acknowledgments

First and foremost, I would like to express my deepest gratitude to God almighty- ALLAH for giving me the strength, blessing and the composure to finish this work. All praises are due to him, words actually will never be enough to express how grateful I am.

I would like to convey my sincere thankfulness to my honorable supervisor Dr. Mohammed Elammari for his guidance, assistance, cordial cooperation and providing valuable advice and constructive criticism in the successful completion of this task.

My heartfelt appreciation deepest regard to my co-supervisor Dr. Kevin Suffern, University of Technology, Sydney–Australia, for his constant guidance, helpful advice and continuous encouragement throughout the progress of this work, without which the work could not have been completed, even my most profound gratitude would not be enough.

I would like to express my sincere acknowledgment to all my academic teachers of Faculty of Information Technology at University of Benghazi, especially honorable Dean Dr. Tawfig Tawill who encouraged me to prepare and submit this thesis in due respect. I also owe a great deal of gratitude to Dr. Omer Elsalabi and Dr. Kenz A. Bozed, the proposal examiners, for their encouragement and insightful comments and durable questions.

Most importantly, I wish to express my appreciation and gratitude to my noble parents for their guidance, sacrifice and continuous encouragement throughout my years of study and my life in general. I also extend my appreciation to my sisters for supporting me spiritually during writing this thesis. I must express my very profound thanks to my faithful aunt Namah, who has inspired me with her hope, love and support.

Ahmad Elhoni

Contents

Symbols

c	speed of light	5
η	index of refraction	5
$\boldsymbol{p}$	ray-object hit point	6
$\boldsymbol{n}$	normal to object surface at $\boldsymbol{p}$	6
θ_i	angle of incidence	6
θ_t	angle of transmission	6
$\boldsymbol{\omega}_\mathrm{o}$	incident ray direction	6
$\boldsymbol{\omega}_\mathrm{r}$, $\boldsymbol{r}$	reflected ray direction	6
$\boldsymbol{\omega}_\mathrm{t}$, $\boldsymbol{t}$	transmitted ray direction	6
η_in	internal index of refraction	6
η_out	external index of refraction	6
π	pi	6
θ_c	critical angle for total internal reflection	7
k_r	Fresnel reflectance	8
k_t	Fresnel transmittance	8
e	specular exponent	10
L	radiance	12
$\boldsymbol{c}_\mathrm{f}$	filter color	12
$\boldsymbol{p}$	a 3D point for evaluating noise	14
$f(x)$	an arbitrary function of the variable x	16
$\in$	interval membership	16
$[a, b]$	a closed interval	16
I	the definite integral of $x \in [a, b]$	16
$\langle I \rangle$	Monte Carlo estimator for I	17
$\boldsymbol{r}_\mathrm{o}$	primary ray	21
dy/dx	derivative of y with respect to x	25
$\partial\eta / \partial x$	partial derivative of η with respect to x	25
C	Luneberg sphere parameter	29
R	Luneberg sphere radius	29
k	inverse sphere parameter	32
c, d	logarithmic spiral parameters	32
θ	logarithmic spiral angle	32
$\boldsymbol{c}$	inverse sphere center	33
$\boldsymbol{e}$	camera eye point in inverse sphere	33
$\boldsymbol{d}$	initial spiral direction	33

$\boldsymbol{p}$	hit point of spiral on sphere surface	33
x_w, y_w, z_w	world coordinates	33
ψ_s	initial spiral direction angle	33
θ_s	initial spiral angle	33
$\boldsymbol{b}, \boldsymbol{b}^{\perp}$	2D orthonormal basis vectors in spiral plane	34
R	inverse sphere radius	35
L	distance between $\boldsymbol{p}$ and $\boldsymbol{e}$	35
S	distance between $\boldsymbol{c}$ and $\boldsymbol{e}$	35
$\boldsymbol{t}$	vector used in calculations	35
$\boldsymbol{i}, \boldsymbol{k}$	unit basis vectors	35
θ	arbitrary angle along a spiral	35
θ_R	angle where spiral hits sphere	35
α, γ	angles used to compute $\boldsymbol{p}$	36
$\boldsymbol{\omega}_o$	tangent vector to spiral at $\boldsymbol{p}$	38
δ	angle used to compute $\boldsymbol{\omega}_o$	38
$+\infty$	positive infinity	39
$\rightarrow$	approach	39
$\downarrow$	approach from above	41
$\pm$	plus or minus	42
a, b	parameters in the η formula for Gaussian spheres	52
η_s	η at the surface of an inverse r sphere	57

Table of Figures

Ray Tracing Dielectrics with Spatially Varying Properties

By

Ahmad Ali Elhoni

Supervisor

Dr. Mohammed M. Elammari

Co-Supervisor

Dr. Kevin G. Suffern

Abstract

Transparent objects are simulated in ray tracing by using reflected and transmitted light rays. We present a simple technique for ray tracing dielectric materials where the filter color inside objects varies with position. The surface reflection and transmission can be specular or glossy. This is based on Whitted-style ray tracing and path tracing. We also ray trace materials where the index of refraction varies with position. This results in curved light paths which in some cases have to be computed numerically. We ray trace Luneberg lenses, and spheres where the index of refraction is proportional to 1 / r, and r is the radial distance from the center. We call these *inverse* spheres. We mathematically analyze the light ray paths inside these spheres, which are logarithmic spirals, and ray trace them from the inside with a fisheye camera. This camera allows rays to be shot in all directions from the location of the camera. We found that the resulting images contained one or more fractals, and our analysis allowed us to understand how the fractals formed. We rendered a number of images to confirm the existence of the fractals.

As a case study, we model a vase in Maya with a high-resolution triangle mesh, and render it in our ray tracer with a variety of materials and ray tracing techniques. These include dielectric and glossy transmitter materials, a rainbow-colored filter color, and path tracing to produce colored caustics.

There is more work to do with the inverse spheres. We want to ray trace them when the index of refraction at the surface is greater than one, and has arbitrarily large values. We also want to place a reflective sphere inside an inverse sphere, and find out what it looks like when we ray trace it with the camera inside and outside.

1 Introduction

1.1 A Historical Overview

Ray tracing is an elegant technique that has its origins in lens making; Carl Friedrich Gauβ traced rays through lenses by hand in the 19th century. Ray tracing algorithms on computers follow the path of infinitesimal rays of light through the scene until they intersect a surface. This approach gives a simple method for finding the first visible object as seen from any particular position and direction. See (Pharr, Humphries, & Jakob, 2017).

Another definition: Ray tracing is a computer graphics rendering technique that simulates geometric optics, where light travels along infinitely thin rays. See (Hecht, 2001). It does this by shooting rays into a scene from a camera, and computing a color for each ray. Ray tracing's great flexibility comes from the fact that it can recursively follow reflected and transmitted rays. This allows it to render reflective and transparent objects, as first discussed in the classic paper Whitted (1980).

1.2 Problem Statements

In the early years of research fields, Ray tracing focused on solving fundamental problems such as determining which objects are visible from a given view point. And then as the scene became more complex and had more detailed materials such as Dielectrics shapes and so on the more effective techniques were needed. See (Pharr, Humphries, & Jakob, 2017).

In CG industry Dielectrics are transparent materials such as glass and acrylic. Air is also a dielectric. When a ray hits the surface of a dielectric object from the inside or the outside, it can be split into two rays where one is reflected, and the other is transmitted through the surface. This creates a binary tree of rays. The transmitted ray is bent away from the original ray's direction, which is called refraction. This is because light travels at a slower speed in a dielectric than it does in air. The index of refraction (ior) of the dielectric determines the speed (Suffern K. , 2007).

1.3 Motivation

Dielectrics can be clear or colored, where the color can be the same at all positions, or it can vary with position, in which case it is called spatially varying. As light passes through a colored dielectric material, some wavelengths are absorbed more

than others, which results in the colored appearance. This process is called color filtering. When the color is spatially varying, it is necessary to sample the color at multiple points along the rays, in order to compute the total amount of filtering for each wavelength.

The index of refraction for most dielectrics is constant, but for some dielectrics, it can also be spatially varying. In this case the rays travel along curved paths, which in some cases have to be computed using numerical techniques.

The surfaces of dielectrics can be perfectly smooth like glass, which results in specular reflection and transmission, where the reflected rays are in the direction of perfect mirror reflection. The surfaces can also be rough, like sand blasted glass where the reflected and transmitted rays are in random directions. This called glossy reflection and transmission.

Imagining the variety of rendering the above physical phenomena in a single master piece image is one of the reasons of a fascinating study area.

1.4 Research Aim and Objectives

This thesis aims to ray trace dielectrics where the spatially varying filter colors are based on noise-based textures (Perlin, 1985). The dielectrics will have specular or glossy reflection and transmission. Also, ray trace a number of spheres where the index of refraction (ior) varies with position. These will be the generalized Luneberg lenses introduced by Suffern and Getto (1991), spheres where the ior is a Gaussian function, and spheres where the ior is the inverse of the radial distance from the center. Then to explore the optical properties of the inverse spheres by ray tracing them with the camera inside, and by plotting interior and exterior ray paths. By doing this, we will use ray tracing to see things that we cannot see any other way. This is a major purpose of the study.

The main objectives of this thesis are the following issues:

- Adding variable filter color support to standard ray tracer.
- Adding variable index of refraction to standard ray tracer.
- Adding texture-based index of refraction to standard ray tracer.
- Adding variable index of refraction to a physically-based ray tracer.
- Comparing existing techniques and related works to our approach, e.g. glossy transmitter and Runge-Kutta methods.

- Study and understand how fractals are formed.
- See how possible in the future to reduce artifacts and speed up process.

1.5 Research Methodology

We have rendered a vase model with a spatially varying filter color, and glossy reflection and transmission using two different techniques, area lighting which is a primitive Ray tracing algorithm. And path tracing to produce soft shadows and colored caustics.

For variable spatially, properties we have used Runge-Kutta numerical technique as in Luneberg lens, and we used analytical solution for the inverse spheres.

Last but not least, solving analytical inverse r spheres equations step by step. Also applying inside camera position for those spheres is worth a try.

1.6 Limitations

The major part of this work was rendering materials with spatially varying filter colors and indices of refraction. This was computationally intensive for a number of reasons, mainly stemming from the 3D lattice noise-based textures we used for the variable filter colors. When the highest spatial frequencies of these were much smaller than the size of the objects, we needed to evaluate the textures many times along each internal ray to adequately sample them. Fortunately, for constant ior we were able use tri-linear-interpolation of the lattice noise points, although for rendering glossy materials, and for path tracing, we needed to use many rays per pixel to reduce the image variance to acceptable amounts. Because of these problems, the final image in Figure 57 has a lot of noise.

We also wanted to use noise-based textures to specify how the ior varies with position. But this situation would be far worse, because the differential equations we would have to solve in Equations (15) require the partial derivatives of the noise functions to be evaluated many times along the internal rays. This means we would have to use tri-cubic interpolation where the computation of each noise point requires the evaluation of 21 cubic Catmull-Rom splines, for each octave. See Suffern (2007) Chapter 30. This would be impractical for a CPU based ray tracer on our single desktop machine, particularly for glossy materials or path tracing. As an indication, Ament et al.

(2014) used a 16 node computing cluster with 2 Intel Xeon X5620 quad core CPUs running at 2.4 GHz, and still had typical rendering times of hours for their images.

GPU acceleration would therefore be something to look at in the future, where the GPUs are used for the shading as well as the ray-object intersections.

1.7 Thesis Organization

The rest of this document is structured as the following: Chapter 2 gives the background information that is needed using Ray tracing dielectrics materials with spatially varying properties. Chapter 3 presents existing and related works. Chapter 4 describes our approach. Chapter 5 presents the implementation and the results of our approach for a simple Ray Tracer software. Chapter 6 shows the extended work done with more complex and realistic Ray Tracer software. Chapter 7 includes the main discussion and gives some perspectives. Finally chapter 8 concludes what is done.

2 Background Material

We discuss here the physics, mathematics, and computer graphics background material that we need for the ray tracing.

2.1 Index of Refraction

Light only travels at a constant speed c, where $c = 2.99 \times 10^7$ m per second, in a perfect vacuum. When light travels through a dielectric (often called a medium), such as the atmosphere of the earth or glass, it slows down because the light interacts with the medium's molecules. We define the absolute index of refraction η of the medium to be the ratio of the speed light in a vacuum c to its speed v in the medium:

$$\eta = c / v.$$

Table 1 shows the index of refraction for some common media.

Medium	Index of Refraction
Perfect vacuum	1.0
Air (1 atm, 20º C)	1.0003
Ice	1.31
Water	1.33
Ethyl alcohol	1.36
Fused quartz	1.46
Acrylic (Plexiglas, Perspex)	1.49
Crown glass	1.52
Polyester resin	1.56
Dense flint glass	1.66
Diamond	2.42

Table 1 *Index of refraction for some common media.*

2.2 Surface Physics

2.2.1 Specular Reflection and Transmission

When a ray from direction $\boldsymbol{\omega}_o$ hits a specular boundary between two transparent media, the reflected ray $\boldsymbol{r}$ makes the same angle θ_i with the normal $\boldsymbol{n}$ as the incoming ray. See Figure 1. This is the law of reflection. Here, we use the standard ray tracing convention that all ray directions point *away* from the hit point $\boldsymbol{p}$. In this case the incoming ray direction is $-\boldsymbol{\omega}_o$.

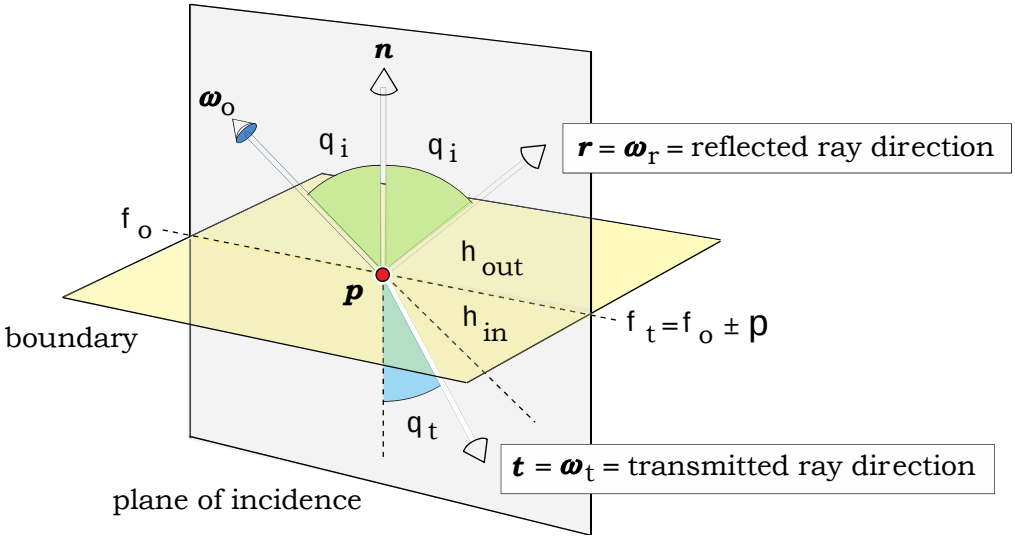


Figure 1. *Reflected and transmitted ray directions at the boundary between two transparent media, Suffern (2007), p. 564.*

The relevant quantity for ray tracing is the ratio η of the two indices of refraction $\eta = \eta_{in}/\eta_{out}$. In Figure 1, θ_t is the *angle of refraction*. This is the angle between the normal direction on the transmitted ray's side of the boundary, and the transmitted ray $\boldsymbol{t}$. The relationship between θ_i and θ_t is known as *Snell's law*:

$$\frac{\sin\theta_i}{\sin\theta_t} = \frac{\eta_{in}}{\eta_{out}} = \eta. \qquad \textbf{Equation 1}$$

Using the law of reflection and Snell's law, we can derive the following expression for the transmitted direction $\boldsymbol{t}$:

$$\boldsymbol{t} = -\frac{1}{\eta}\boldsymbol{\omega}_o - \left(\cos\theta_t - \frac{1}{\eta}\cos\theta_i\right)\boldsymbol{n}, \qquad \textbf{Equation 2}$$

where

$$\cos\theta_i = \boldsymbol{n} \bullet \boldsymbol{\omega}_o$$

and

$$\cos\theta_t = (1 - \frac{1}{\eta^2}\sin^2\theta_i)^{1/2}. \qquad \textbf{Equation 3}$$

See Glassner (1989) and Shirley and Marschner (2009). By construction, $\boldsymbol{t}$ is a unit vector.

When light passes from a medium with a smaller η to a medium with a larger η, $\boldsymbol{t}$ is bent towards the normal direction at the hit point (see Figure 4.2(a)). But when light passes from a medium with a larger η to a medium with a smaller η, $\boldsymbol{t}$ is bent away from the normal direction as shown in Figure 2(b).

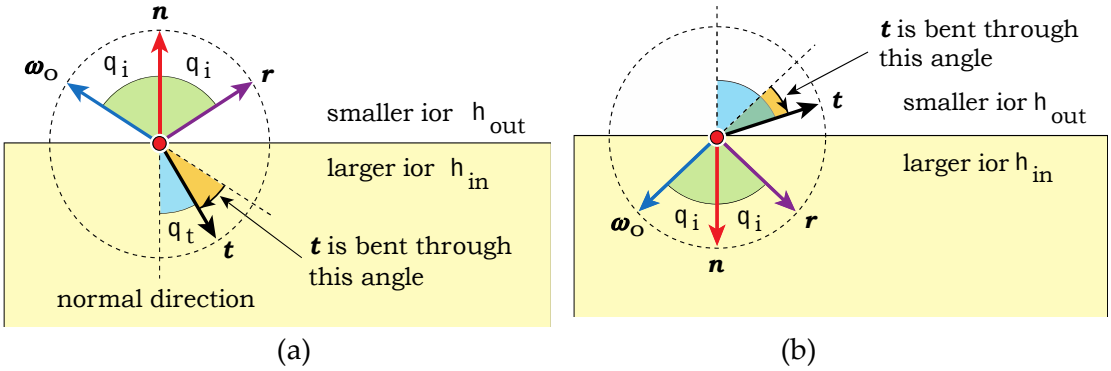


Figure 2 *(a) Direction change of transmitted ray* **t** *when* $\eta > 1$*, (b) Direction change of* **t** *when* $\eta < 1$.

This can lead to the optical phenomenon called *total internal reflection*, as illustrated in Figure 3. As the incident angle θ_i increases, it can reach a critical angle θ_c where the transmitted ray is parallel the surface. This is the situation in Figure 3(a) where $\boldsymbol{t}$ is a dashed line because it contains no energy. If $\theta_i > \theta_c$, there is no transmitted ray, and the boundary becomes a perfect specular reflector.

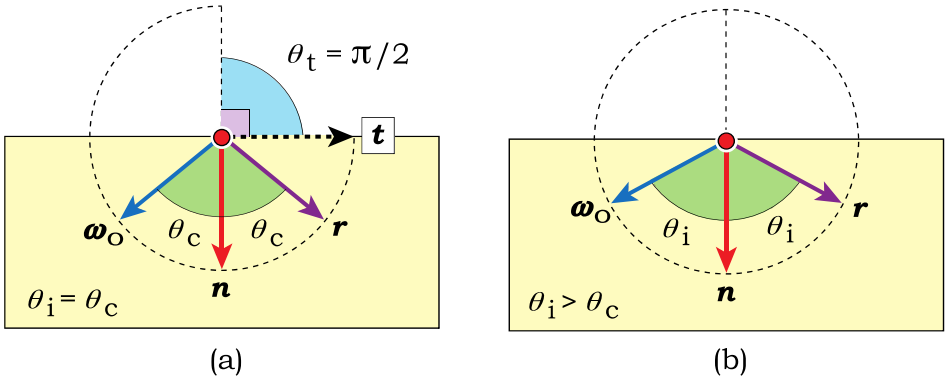


Figure 3. *Total internal reflection: (a)* $\theta_i = \theta_c$ *; (b)* $\theta_i > \theta_c$.

In Figure 2(a), the energy in the incident ray is split between ***r*** and ***t***, but as the direction of ***t*** approaches the boundary, the energy it carries decreases, while the energy carried in the reflected ray ***r*** increases. By the time ***t*** is parallel to the boundary, there is no energy in it. When total internal reflection occurs, all of the energy from the incident ray goes into the reflected ray. The Fresnel equations, which are discussed below, specify the exact energy split between ***t*** and ***r*** as a function of θ_i and η.

To produce correct images, total internal reflection must be taken into account when we ray trace transparent objects. We have to test for this condition at each ray–transparent-object hit point. Fortunately we don't have to compute θ_c; total internal reflection occurs when the expression inside the square root in Equation (3) is less than zero. In this case the expression for ***t*** would contain a complex number, which means that ***t*** doesn't exist as a real ray. The test is therefore

$$1 - \frac{1}{\eta^2} \sin^2 \theta_i < 0 \qquad \textbf{Equation 4}$$

2.2.2 Fresnel Equations

A simple model of transparent materials in ray tracing uses constant reflection and transmission coefficients k_r and k_t, where $k_r + k_t = 1.0$. The French physicist and mathematician Augustin-Jean Fresnel (1788-1827) discovered how these quantities vary with the incidence angle θ_i. These involve polarized light (Hecht, 2001):

$$r_{\parallel} = \frac{\eta \cos\theta_i - \cos\theta_t}{\eta \cos\theta_i + \cos\theta_t},$$

Equation 5

$$r_{\perp} = \frac{\cos\theta_i - \eta\cos\theta_t}{\cos\theta_i + \eta\cos\theta_t}$$

where $\eta = \eta_{\text{in}}/\eta_{\text{out}}$ and $r_{\parallel}$ and $r_{\perp}$ are the reflected amplitudes of light waves polarized parallel and perpendicular to the boundary respectively. For un-polarized light, the *Fresnel reflectance* k_{r} is given by

$$k_r = \frac{1}{2}(r_{\text{p}}^2 + r_{\perp}^2). \qquad \textbf{Equation 6}$$

By conversion of energy, the *Fresnel transmittance* k_{t} is

$$k_t = 1 - k_r. \qquad \textbf{Equation 7}$$

Figure 4 shows plots of k_{r} and k_{t} for glass with η = 1.4. Here, k_{r} is 4% at normal incidence, and rises to 100% grazing incidence. In fact, all smooth dielectrics and metals become perfect mirrors at grazing incidence.

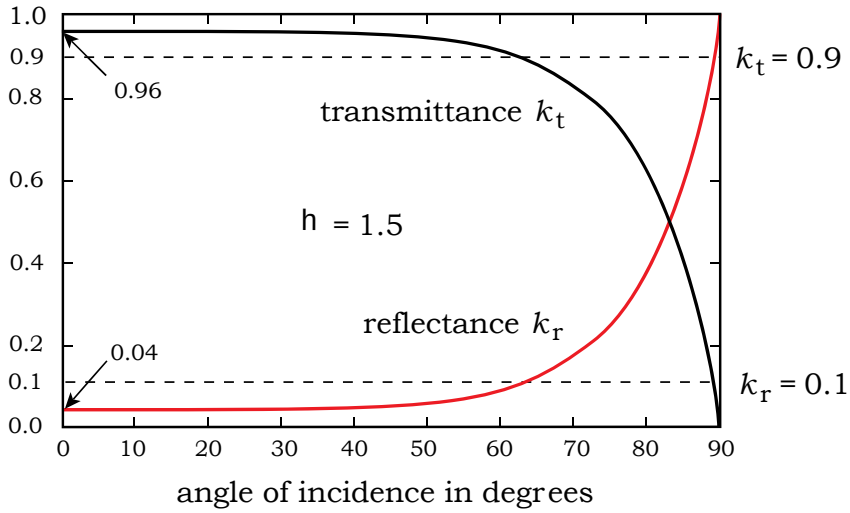


Figure 4. *Fresnel reflectance and transmittance for glass as a function of incidence angle, Suffern (2007), p. 596.*

2.2.3 Glossy Reflection and Transmission

The reflection and transmission of real materials can vary from diffuse to specular as illustrated in Figure 5. A diffuse reflector is like matte paint, where the light is scattered in all directions.

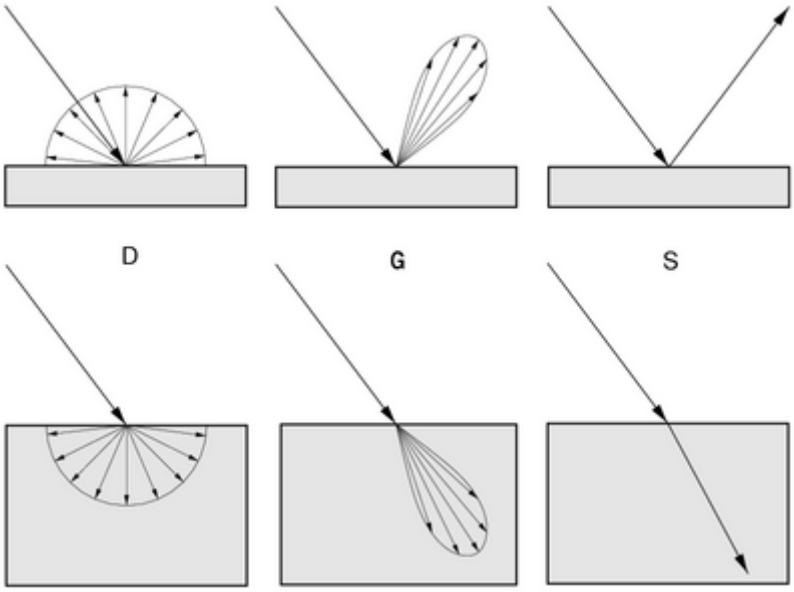


Figure 5. *Reflectance and transmission: diffuse on the left, glossy in the middle, and specular on the right.*

We model glossy reflection by using random directions for the reflected rays measured from the direction of mirror reflection ***r***. See Figure 6.

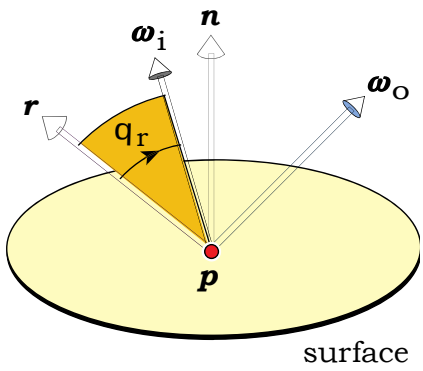


Figure 6. *The reflected ray direction* ω_i *makes an angle* θ_r *with the direction of mirror reflection* $\mathbf{r}$, *Suffern (2007), p. 530.*

The density *d* in solid angle of these rays is given by

$$d = (\cos \theta_r)^e \qquad \textbf{Equation 8}$$

where *e* is the specular exponent. When *e* = 1.0 we have diffuse reflection. Note that this is modeled with a cosine distribution instead of a constant density. As *e* increases, the surfaces become less glossy, and in the limit *e* –> ∞, they become specular. These distributions of ray directions are known as *Phong lobes*. A feature of this simple empirical model is that a fraction of the Phong lobe is always below the surface, and for

incoming rays at grazing incidence, the fraction is one half. Any rays with directions below the surface have to be killed, which results in limb darkening, as shown in Figure 7. The limb darkening is however, physically correct.

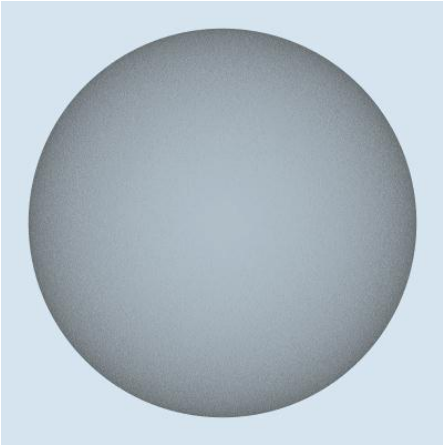


Figure 7. *Diffuse sphere (*e *= 1.0) rendered with 100 rays per pixel.*

We model diffuse and glossy transmission in an analogous way, but here we don't kill the rays below the surface, because they are refracted, and then internally transmitted (the ray ***t*** in Figure 1). This allows us to render dielectric materials that look like frosted glass, as shown in Figure 8. More sophisticated and physically accurate models of glossy reflection and transmission have been developed based on *micro facets*, starting with Blinn (1977).

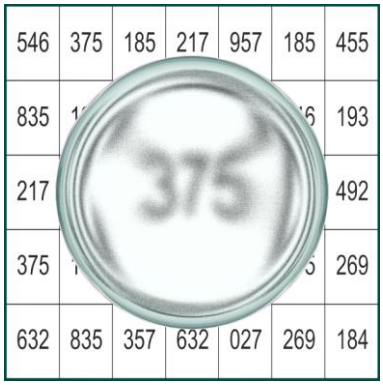


Figure 8. *Glossy transmission material rendered with e = 10000.0.*

2.3 Color Filtering

When light passes through dielectrics some of it can be absorbed by the molecules. The dielectric will be colored if the amount of absorption depends on the

wavelength of the light. According to the *Beer-Lambert law*, the transmitted radiance $L(d)$ can be written as:

$$L(d) = \boldsymbol{c}_{\mathrm{f}}^{d} L_{\mathrm{o}}, \qquad \textbf{Equation 9}$$

where d is the distance traveled in the dielectric material, L_{o} is the value of L when $d = 0$, and $\boldsymbol{c}_{\mathrm{f}}$ is the filter color, Suffern (2007), p. 596.

We use a simple RGB color for all colors in the ray tracer, where white is (1.0, 1.0, 1.0). If $\boldsymbol{c}_{\mathrm{f}}$ is white, there is no color filtering (and the dielectric is clear), but if any color component is less than 1.0, it's value decreases exponentially with the distance travelled through the dielectric. This is illustrated in Figure 9.

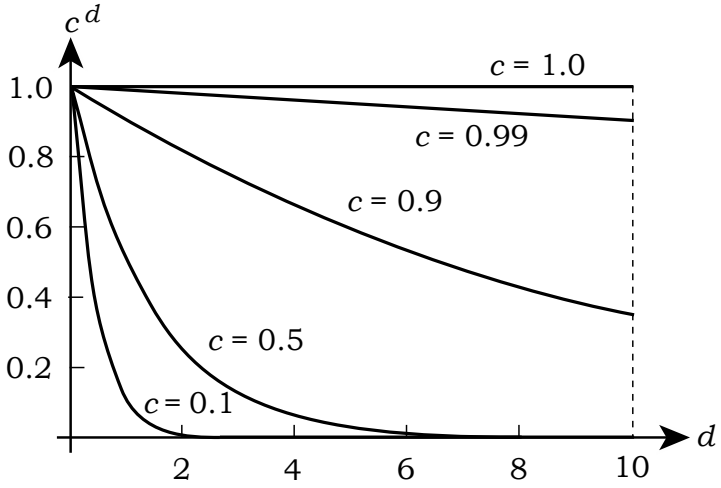


Figure 9. *Plots of* c^{d} *for various values of* c, *Suffern (2007) p. 598.*

A good way to demonstrate beer's Law is to ray trace a concave lens as shown in Figure 10. Here, the thickness of the lens at the rim is 4.0, but at the center it's only 0.1.

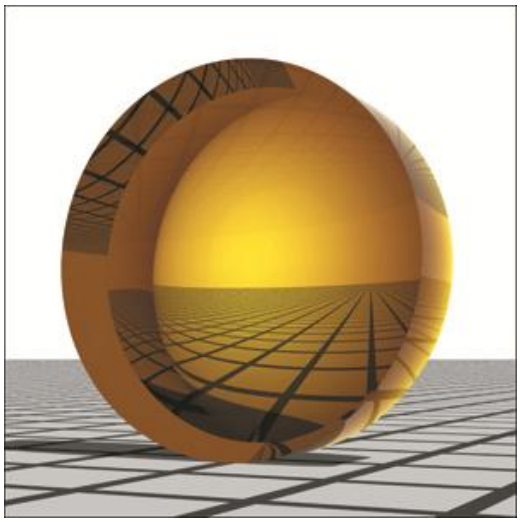


Figure 10. *A concave lens rendered with* $\mathbf{c}_f$ *= (0.65, 0.45, 0.0), Suffern (2007), p. 608.*

2.4 Noise-Based Textures

2.4.1 Lattice Noise

A characteristic of natural textures is their randomness. They may contain structures or colors that repeat, but the exact details are always different. Stones, woods, and clouds are examples. We can simulate many of these textures with functions built from pseudo-random numbers. These are known as *noise functions*. The original functions were invented by Ken Perlin in 1985, and have been used extensively for image synthesis ever since. To produce good 3D noise-based textures, we need to do a number of things. The first is to produce a smoothly varying 3D random function that has a known range, and does not repeat. Perlin (1985) used a simulated infinite 3D lattice of random numbers in the range (-1.0, +1.0) at the corners of unit cubes. See Figure 11.

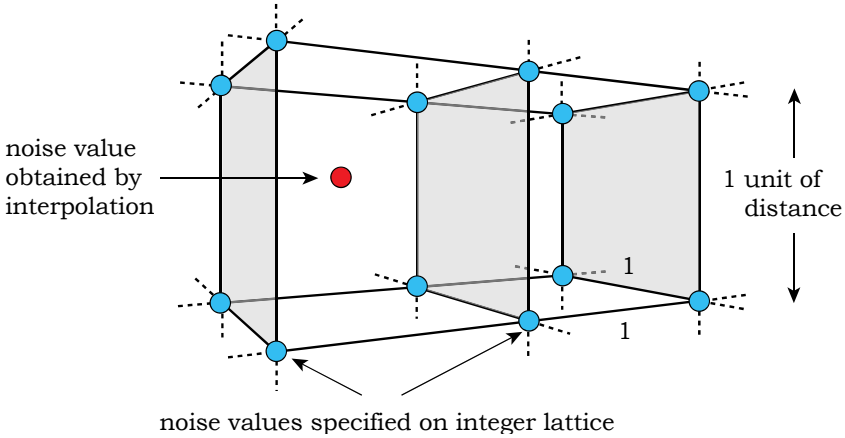
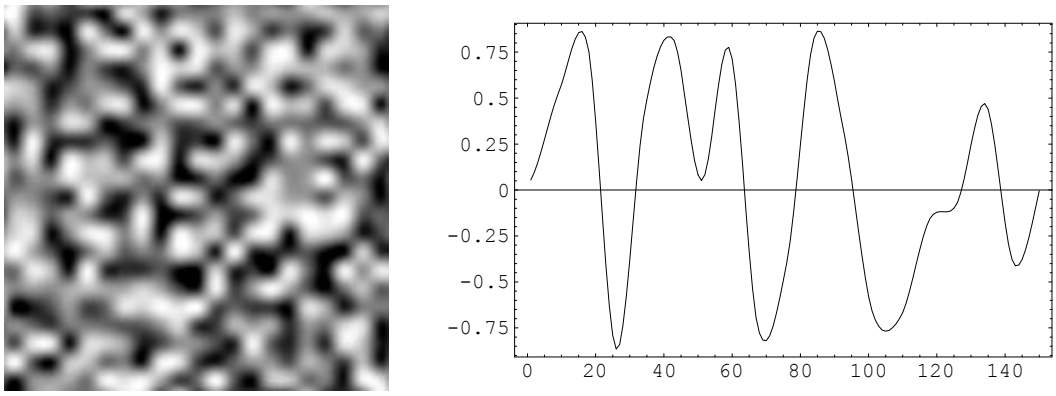


Figure 11. *3D infinite lattice, Suffern (2007), p. 694.*

These values are interpolated using tri-cubic interpolation to give the value of the noise function at any point in space. The result is known as a *lattice noise*. This is just a quick overview of a process that involves the cell where the point is, its 26 neighbors: face, edge, and vertex; and 21 cubic Catmull-Rom splines. The best places to read about these are Peachy (2003) and Suffern (2007), Section 31.2.2.

Figure 12(a) shows a 2D image of the lattice noise, and Figure 12(b) shows a 1D plot of the noise along the center row of pixels in part (a). Part (a) shows that this noise function is not suitable for texturing, because the cellular structure of the cubes is evident.



(a) (b)

Figure 12. *(a) 2D slice of tri-cubic interpolated lattice noise; (b) plot of the lattice noise along the middle row of pixels in (a).*

4.4.2 Spectral Synthesis

It's far more useful to use *spectral synthesis*, which involves a weighted sum of the noise function and its octaves. If $\boldsymbol{p}$ is a point where we want to evaluate the noise, the *fractal sum* function is defined as

$$fractal\_sum(\boldsymbol{p}) = \sum_{j=0}^{n-1} \frac{noise(2^j * \boldsymbol{p})}{2^j},$$

Equation 10

where each term has a smaller amplitude by a factor of two, but changes twice as fast with distance. Figure 13, which is based on Apodaca and Gritz (2000), p. 252, illustrates this process.

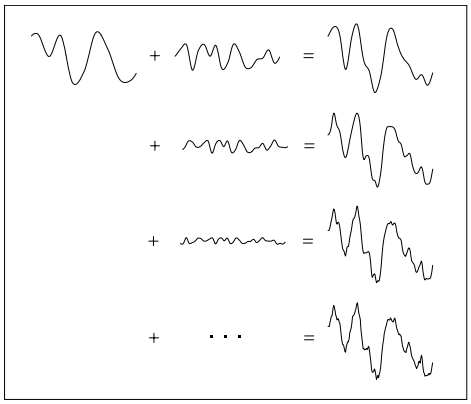


Figure 13. *Terms in the **fractal_sum** function, and their sums.*

Figure 14(a) shows a single octave, which is just cubic noise; part (b) shows two octaves where there is still some evidence of the cubic lattice; part (c) shows 8 octaves, where there is still some evidence, but it's not objectionable.

(a) (b) (c)

Figure 14. *2D cross sections of the **fractal_sum** function with 1 octave in (a), 2 octaves in (b), and 8 octaves in (c).*

2.4.3 Wrapped Noise Textures

Wrapped textures are created by increasing the range of a lattice noise, and using the Standard C Library function floor to create discontinuities in the returned value. Because this is hard to explain in words, it's best just to read the pseudo-code below. This results in ridges in the texture, as illustrated in Figure 15. The details are in Suffern (2007), p. 720. In chapter 5 we will ray trace a transparent object where the filter color is a wrapped texture, because this has not been done before.

```
RGBColor wrappedtexture::get_color(hit_point) {
      noise = multiplier * noise_function(hit_point)
      value = noise - floor(noise)
      return (value * color)
```

```
}
```

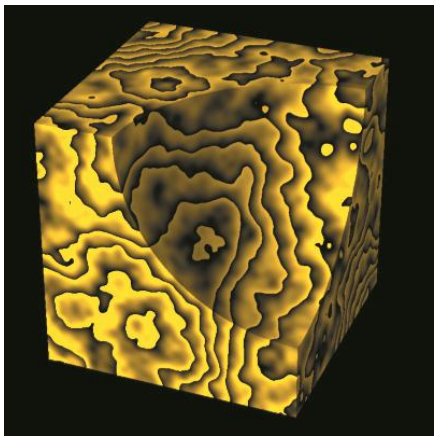


Figure 15. *Example of a wrapped texture.*

4.5 Monte Carlo Integration

When we ray trace scenes with area lights, or materials with glossy reflection or transmission, we need to numerically estimate the values of multi-dimensional integrals. This is a result of solving the rendering equation. The most efficient way to do this is to use random numbers. The technique is called Monte Carlo integration, and was first discussed by Lord Kelvin in 1901 and then independently rediscovered by Enrico Fermi, John von Neumann, and Stanislaw Ulam in the 1940's. The book Handscomb and Hammersley (1964) is a good introduction to the subject.

Below is a brief introduction using a one-dimensional example. Figure 16 shows part of a function $f(x)$ for $x \in [a, b]$, where the definite integral is the shaded area under the graph.

$$I = \int_a^b f(x)\,dx$$

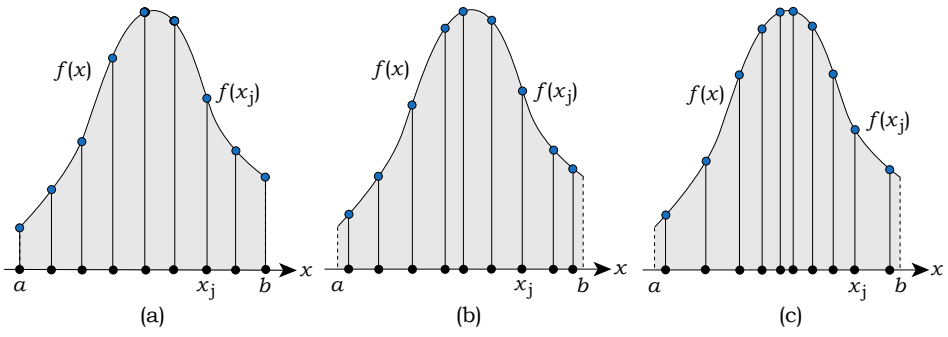


Figure 16. *(a) equally spaced sample points; (b) uniformly distributed random sample points; (c) randomly distributed sample points for importance sampling.*

This figure shows three techniques for numerically estimating its value with sample points. In part (a), the samples are equally spaced in the x direction. Joining the blue sample points with straight lines creates a series of trapezoids whose area can be summed to estimate the integral. The more sample points we use, the more accurate the result. This is the trapezoidal technique from traditional numerical quadrature. The problem with this and related techniques is that the results get worse as the dimensionality of the integrals increases. In contrast, Monte Carlo techniques get more accurate as the dimensionality increases.

In Figure 16(b), the samples use uniformly distributed random values of x in the interval $[a, b]$. The *Monte Carlo estimator* for the integral I, denoted by $\langle I \rangle$, is

$$\langle I \rangle = \frac{b-a}{n} \sum_{j=1}^{n} f(x_j). \qquad \textbf{Equation 11}$$

Here, we have used the *mean value theorem*, as discussed in most calculus texts.

For a given number of samples n, we can get a more accurate estimate by making the density of the points match the shape of the function as closely as we can. This is known as *importance sampling*, and is illustrated in Figure 16(c). The estimator in this case can be written using a *probability distribution function* or pdf for short:

$$\langle I \rangle = \frac{1}{n} \sum_{j=1}^{n} \frac{f(x_j)}{p(x_j)}. \qquad \textbf{Equation 12}$$

The closer the pdf matches the function, the better the results will be. The problem with ray tracing is that, although the points are 3D, the integrals can have arbitrarily high dimensions, and we don't know what the functions are. But usually we know something about them. For example, if there is diffuse reflection (from matte surfaces), the function will involve cos (θ_i), and we can use that.

Regardless of the sampling technique we use, the resulting images will generally be correct provided we use enough sampling points. Importance sampling is just more efficient than uniformly distributed random sampling. Errors show up as noise in the images. In contrast, if we used equally spaced points, the errors would be artifacts in the images, which are much more objectionable than noise.

3 Literature Review

Most of the papers discussed here presented the original ray tracing and related research material that our work is based on. These papers are sorted in a relevant to the evolution of ray tracing.

3.1 Whitted (1980) An Improved Illumination Model for Shaded Display

Before Whitted's landmark paper (Whitted, 1980), ray tracing was only used for ray casting, where the rays stopped when they hit an object. This allowed the direct illumination from light sources to be rendered, which existing scan-line algorithms could also render. Whitted presented an algorithm that allowed part of the global illumination in a scene to be rendered. For transparent materials, a binary tree of rays is recursively constructed and traced for each pixel. This starts from the viewer to the first surface intersected, and from there to other surfaces. This allowed the ray tracer to accurately render reflections, shadows, and transparency with refraction. These effects could not be rendered by scan-line algorithms.

3.2 Kajiya (1986) The Rendering Equation

The rendering equation, as introduced by Kajiya (1986) was Chandrasekhar's equation of radiative transfer, originally published in Chandrasekhar (1960), but re-written in terms of intensity. This is a Fredholm integral equation of the second kind which has an infinite series expansion solution. Kajiya showed that the first term represents ray casting, multiple terms represent path tracing and radiosity, and a double series solution with multiple terms represents Whitted ray tracing. This is another landmark paper because it provided a unified theoretical foundation for all these rendering algorithms. Now, we express the rendering equation in terms of radiance, because this is the fundamental radiometric quantity transmitted by rays.

3.3 Perlin (1985) An Image Synthesizer and Peachy (1985) Solid Texturing of Complex Surfaces

Perlin (1985) and Peachey (1985) simultaneously introduced 3D procedural textures to computer graphics, and the use of Fourier synthesis for texture generation. Peachey used sums of sinusoids, but Perlin used sums of 3D lattice noise functions. He also introduced a turbulence function which is a sum of the absolute values of the noise

functions. Both authors used their techniques to render a variety of 3D textures. We used Perlin's noise function to define spatially varying filter colors for transparent objects.

3.4 Evans and Rosenquist (1985) $F = ma$ Optics

These authors showed that *Fermat's Principle*, see for example Hecht (2001), can be used to derive equations of the form $F = ma$ (Newton's second law of motion) for determining the shapes of light paths in media with spatially varying indices of refraction. A big advantage of this technique over the traditional method of using the non-linear eikonal equation is that the resulting equations are linear. This makes them much simpler to solve analytically, or if that's not possible, they can be solved with standard numerical techniques such as Runge-Kutta integration (see Section 4.6).

3.5 Suffern and Getto (1991) Ray Tracing Gradient Index Lenses

In spite of the advantages of $F = ma$ optics, this paper seems to be the only publication to have used it for ray tracing. These authors ray traced images of two families of objects with spatially varying indices of refraction: generalized Luneburg lenses, which are spheres, and gradient index rod lenses, which are circular cylinders. The equations of motion of light rays inside the lenses were solved analytically or numerically.

3.6 Lee and Uselton (1991) A Body Color Model: Light Absorption Through Translucent Media

These authors discussed the absorption of light by transparent media where the amount of absorption depends on the wavelength of the light. They derived the equation for the Beer-Lambert law of absorption for homogeneous media, which we discussed in Section 2.3. They also discussed absorption in inhomogeneous media.

3.7 Ament, Bergman, and Weiskopt (2014) Refractive Radiative Transfer Equation

This recent paper introduced a refractive radiative transfer equation for media with a spatially varying index of refraction. Their work was based on Hamiltonian dynamics to describe light propagation along curved paths. They implemented their work as an extension to the photon mapping algorithm introduced by Jensen (2001).

4 Techniques

4.1 Overview

In this chapter we discuss practical techniques for rendering dielectrics with spatially varying indices of refraction, and textured color filtering. Figure 17 illustrates the various types of dielectrics and the people who originally studied, or ray traced them.

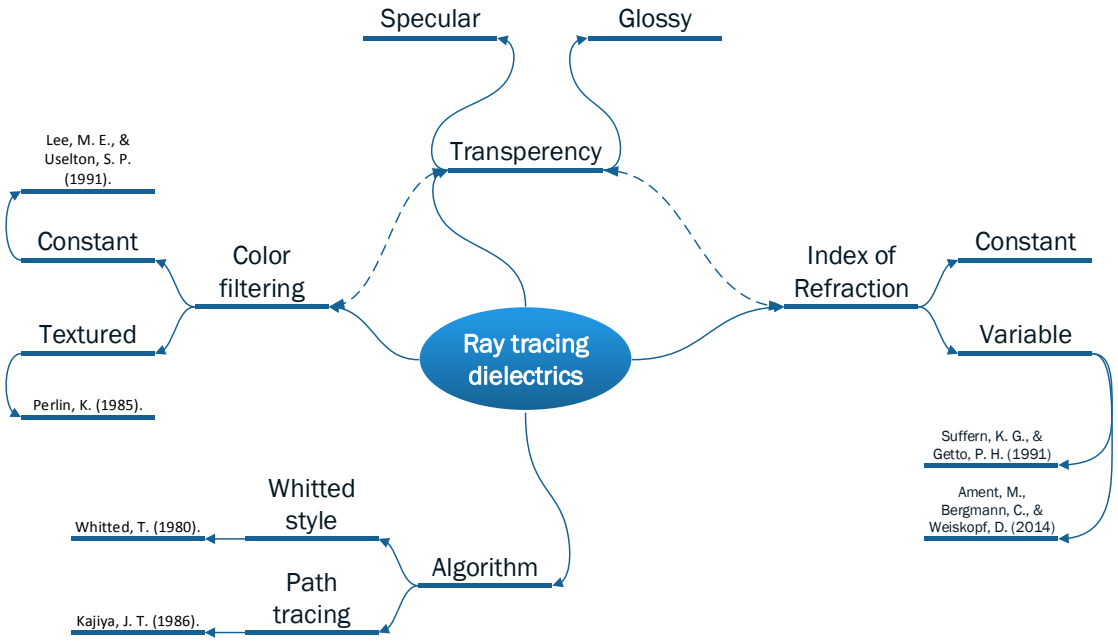


Figure 17. *Types of dielectric materials.*

4.2 Whitted Ray Tracing

In Whitted ray tracing, reflected and transmitted rays are followed recursively until a specified maximum recursion depth is reached, or some other condition terminates them. This is illustrated in Figure 18, where the scene consists of two transparent objects, an opaque non-reflective object, one other object, and is ray traced with a maximum depth of four. Here, $\boldsymbol{r}_o$ is the primary ray, which starts at the camera, and is at depth zero.

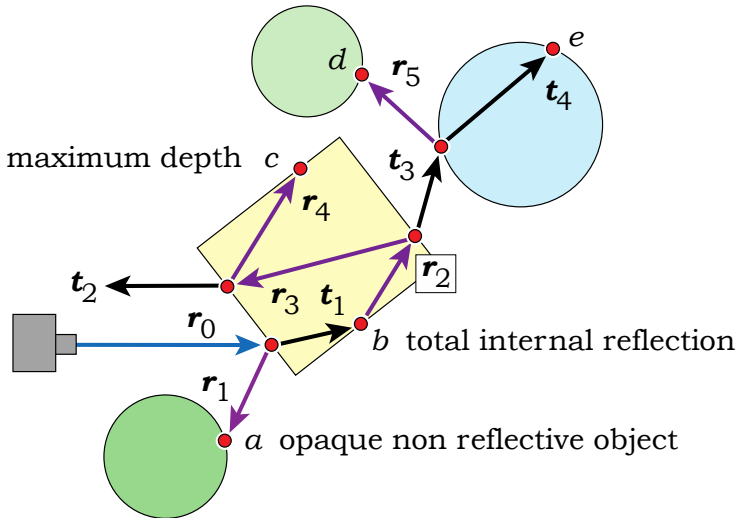


Figure 18. *Transparent objects with reflected and transmitted rays, Suffern (2007), p. 569.*

This results in a binary tree of rays, as shown in Figure 19, which can result in long rendering times. In practice, branches are often terminated before they reach the maximum depth. In Figures 18 and 19 this has happened when one ray has hit a non-reflective object, and another has encountered total internal reflection.

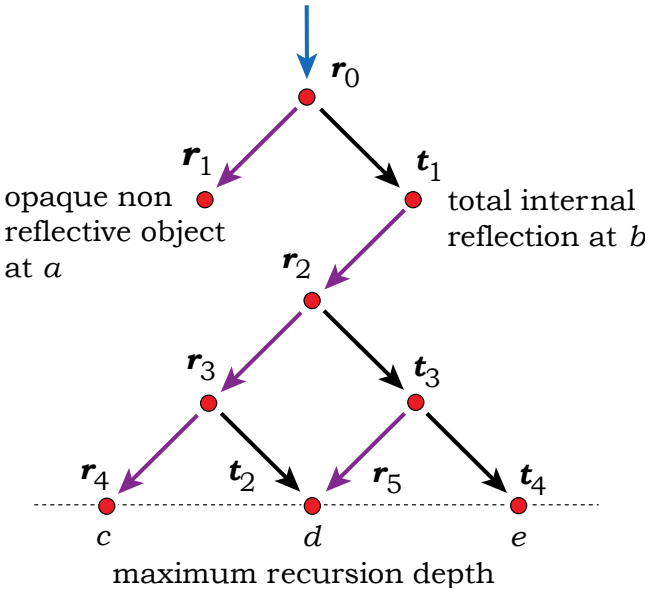
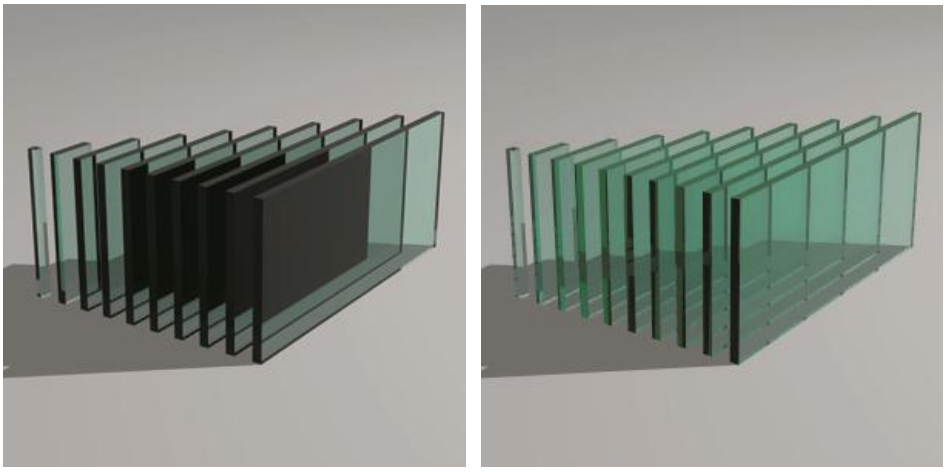


Figure 19. *The ray tree that corresponds to Figure 18 , Suffern (2007), p. 569.*

As an example, Figure 20(a) shows a collection of glass blocks ray traced with a maximum depth of 4, which is too low for all the rays to get though, as shown by the black areas. In Figure 20(b) the maximum depth is 15, which seems to be sufficient, but this took a long time to render.



(a) (b)

Figure 20. *Glass blocks with η = 1.5, color filtering, and the Fresnel Equations for reflection and transmission: (a) max depth = 4, (b) max depth = 15 , Suffern (2007), pp. 635 and 616.*

4.3 Path Tracing

Whitted ray tracing can only simulate certain parts of the light transport in scenes. For example, light can be reflected from a mirror onto a wall, or concentrated by a magnifying glass onto a piece of paper. Although Whitted ray tracing can follow the rays that are reflected and transmitted through dielectrics, it cannot render the light that is reflected from the surfaces that these rays hit. These are called *caustics*. It also cannot simulate diffuse to diffuse light transport. These are all parts of indirect illumination, which is also called global illumination.

Fortunately, direct and indirect illumination can be rendered with *path tracing*, in scenes with area lights. Unlike point lights, which cannot exist, area lights have finite surface area. Path tracing is a conceptually simple brute-force technique which works as illustrated in Figure 21. Each ray is recursively traced through the scene until it either: reaches a light source, or the maximum recursion depth is reached, or it leaves the scene. The smaller the light sources are, the more rays are needed to reduce the noise to an acceptable level.

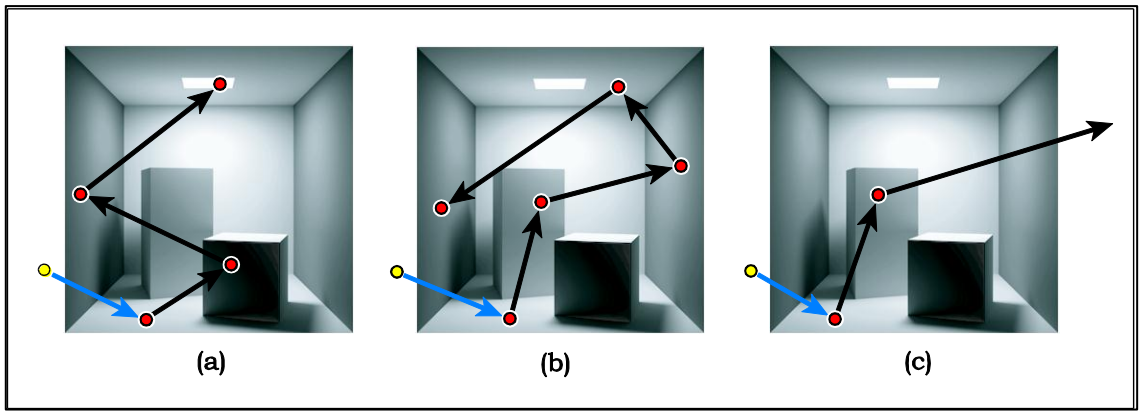


Figure 21. *Gray scale image of the Cornell box scene originally path traced in color by Steve Parker with 100498 rays per pixel. (a) ray hits the light, which is the only source of radiance, (b) ray terminates at the maximum recursion depth of five, (c) ray leaves the scene. Suffern (2007), p. 544.*

We use path tracing to render a vase with a colored glossy transmitter material because the colored caustics add extra realism to the images. Photon mapping, developed by Jensen (2001) is more efficient than path tracing for rendering caustics, and works with point lights as well as area lights.

4.4 Spatially Varying Filter Color

The formula for color filtering in Equation (9) only applies when the filter color is constant. When the color varies with position, we have to sample it along the ray as illustrated in Figure 22.

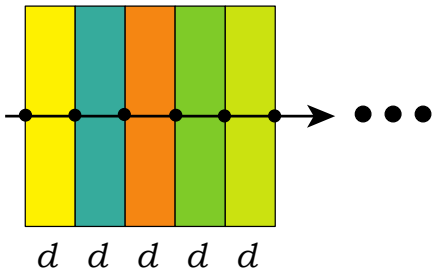


Figure 22. *When we sample a spatially varying texture along a ray and use Equation (9) for the filter color, we approximate the texture as slabs of constant filter color. This figure is schematic only; the 3D filter color in the surrounding dielectric can vary in any way.*

The formula for the filtered color is in Equation (13) where d is the distance along the ray between the sample points.

$$\boldsymbol{c} = \prod_{j=1}^{n} \boldsymbol{c}_f \left(\boldsymbol{p}_j \right)^d \qquad \text{Equation 13}$$

4.5 Spatially Varying Index of Refraction

When the index of refraction of a dielectric varies with position, the rays follow curved paths, as illustrated in Figure 23. In simple cases such as Luneberg lenses and mirage formation above hot road surfaces, the paths can be computed analytically (Evans and Rosenquist, 1985). In other cases, the ray paths have to be computed numerically, for which we will use the standard Runge-Kutta technique, as described below.

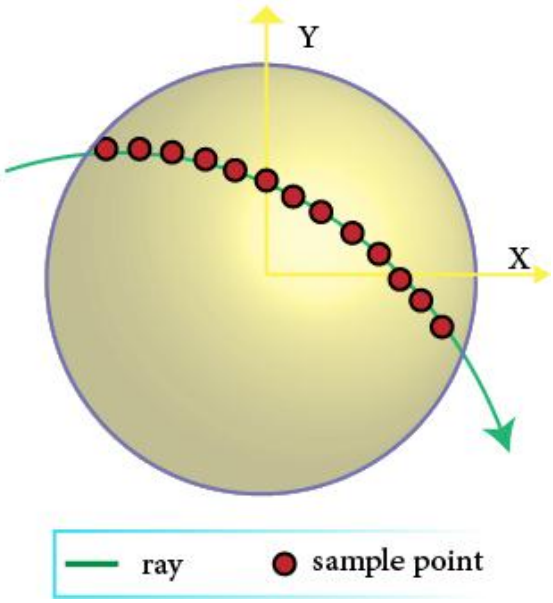


Figure 23. *A curved ray path through a dielectric with a spatially varying index of refraction.*

To compute the color filtering, we approximate the ior along the ray as slabs with constant η between the sample points. See Figure 24. In this case the sample points are not equally spaced in distance along the ray, because of the numerical integration.

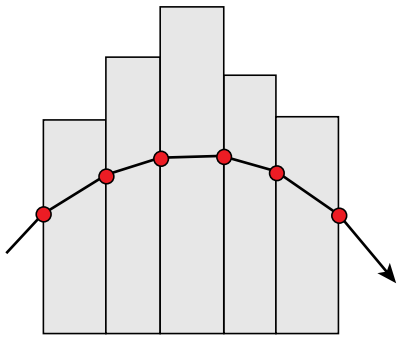


Figure 24. *Here, the heights of the slabs represent the constant values of the ior between the sample points on the ray. This is another schematic diagram.*

We use the formula in Equation (14) for the color filtering, where d_j is the distance between the points j - 1 and j. The η^2 factors are due to the refraction of light as the ior changes at each sample point. See for example Suffern (2007), p. 568.

$$\boldsymbol{c} = \prod_{j=1}^{n} \frac{\eta_{j-1}^2}{\eta_j^2} \boldsymbol{c}_f^{d_j} \qquad \text{Equation 14}$$

4.6 Runge-Kutta Integration

Runge-Kutta integration is a common numerical technique for obtaining approximate solutions to systems of first-order differential equations, given a set of initial conditions. For the single equation $dy/dx = f(x, y)$ with initial condition $y(x_0) = y_0$, the fourth-order Runge-Kutta equations are:

$$y_{n+1} = y_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4),$$

$$k_1 = f(x_n, y_n),$$

$$k_2 = f\left(x_n + \tfrac{1}{2}h, y_n + \tfrac{1}{2}hk_1\right),$$

$$k_3 = f(x_n + \tfrac{1}{2}h, y_n + \tfrac{1}{2}hk_2),$$

$$k_4 = f(x_n + h, y_n + hk_3),$$

where h is the (constant) step size in x. See for example Ralston and Rabinowitz (2001) and Zill, Wright, and Cullen (2012), p. 368. Although h can vary from step to step, we will use a constant value.

The equations to be solved for points (x, y, z) along the curved ray paths can be written as the following three second order differential equations where a is an affine parameter, and $\eta(x, y, z)$ is the ior. See Suffern and Getto (1991).

$$\frac{d^2x}{da^2} = \eta(x,y,z)\frac{\partial \eta}{\partial x},$$

$$\frac{d^2y}{da^2} = \eta(x,y,z)\frac{\partial \eta}{\partial y}, \qquad \text{Equation 15}$$

$$\frac{d^2z}{da^2} = \eta(x,y,z)\frac{\partial \eta}{\partial z}.$$

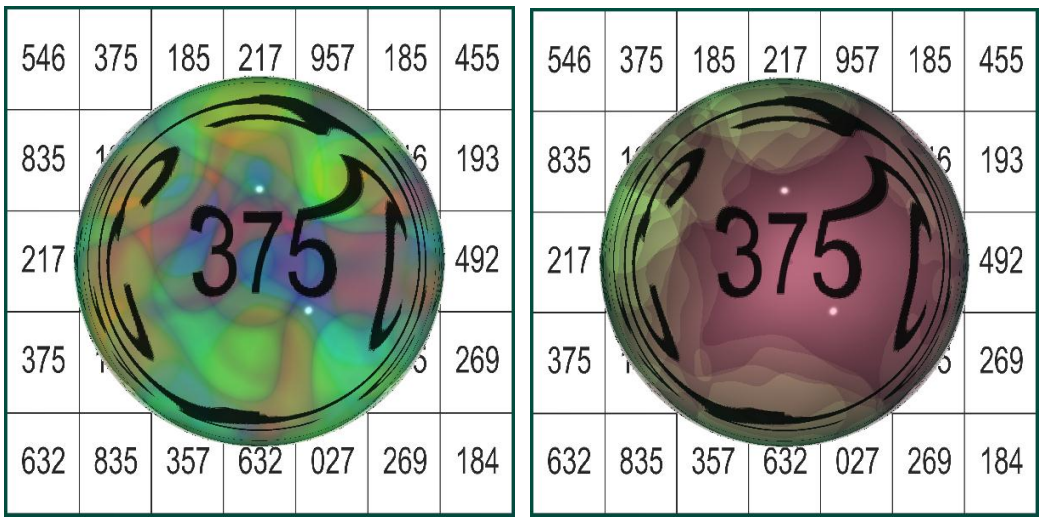
These can be written as 6 first order differential equations which we will solve with a fourth order Runge-Kutta technique.

5 Simple Ray Tracer

For the results presented here, we used the ray tracer discussed in Suffern (2007) and then further developed by Rosser (2012). This later version runs in Visual Studio, has a good user interface, allows arbitrarily sized images to be rendered, is multi-threaded, uses smart pointers and reference counting so there are no memory leaks, and allows images to be saved in a number of file formats. We have included the Data Flow diagram in the Appendix. Chapter 6 shows a more realistic render aka LuxRender to produce more authentic some images, see for latest version www.luxcorerender.org

5.1 Textured filter color

Figure 25(a) shows a sphere rendered with a fractal sum filter color as defined in Equation (10), where the rainbow-like colors come from evaluating Equation (5.1). In Figure 25(b), the filter color is a wrapped noise texture, which shows the characteristic ridges of this type of noise function. See Figure 15.



(a) (b)

Figure 25. *(a) Dielectric sphere with a fractal sum rainbow colored filter, (b) a wrapped noise texture.*

Figure 26 shows a number of objects rendered using the techniques discussed above. The orange sphere has a wrapped texture; the rectangular box is a dielectric with a blue-green filter; the pink cylinder has a glossy reflective material; the large sphere has a rainbow texture color filter; the small sphere has a glossy dielectric material with olive-green color filtering. The spheres show some perspective distortion. This image was rendered using path tracing with max depth = 10 and approximate 64 samples per pixel.

A large hollow sphere with an emissive material on its inside surface acts as the light source. This results in the soft shadows on the plane. It also results in the colored caustics on the plane (which are difficult to see.)

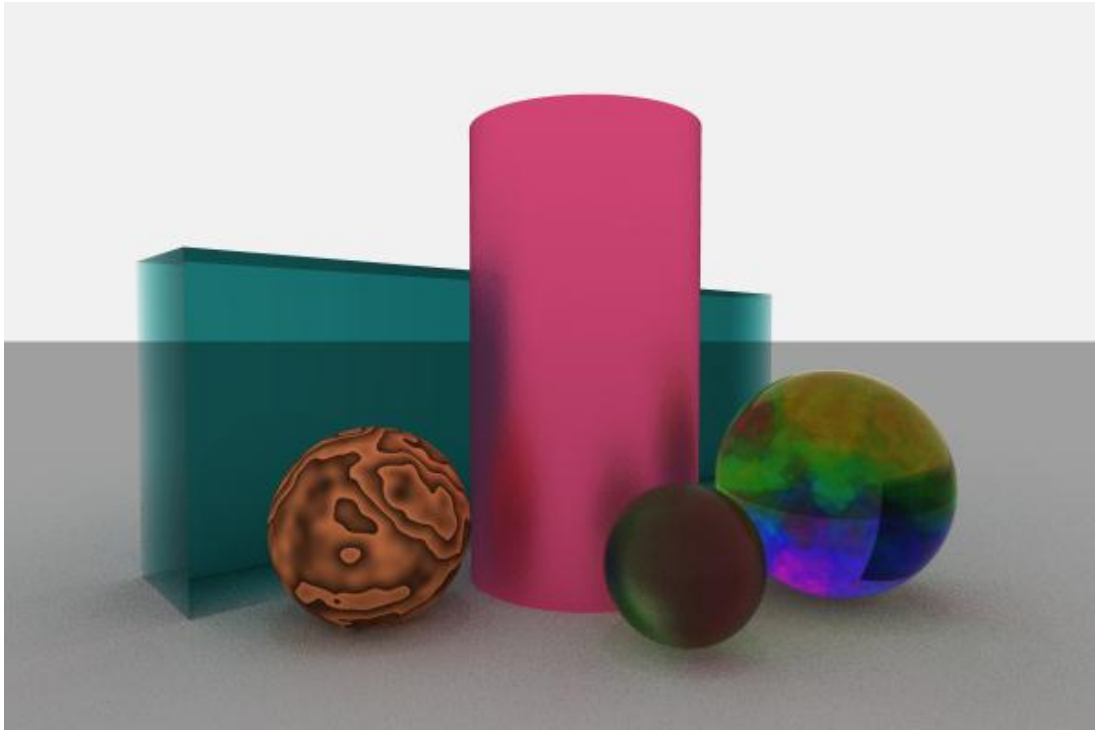
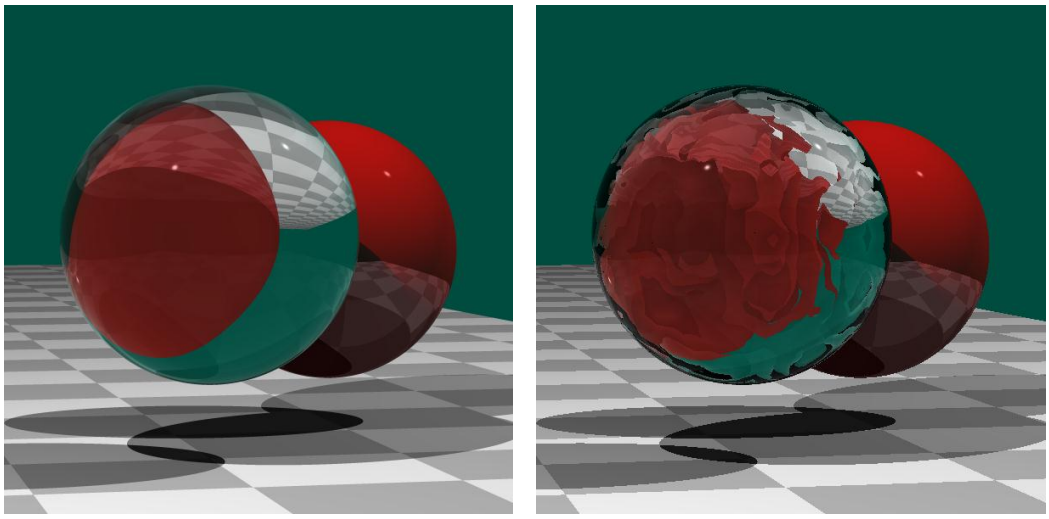


Figure 26. *Path traced objects, with different shaders.*

5.2 Spatially Varying Dielectrics

5.2.1 Surface Variation

Figure 27(a) is a reference image of a dielectric sphere with η = 1.5, and a reflective sphere. In Figure 27(b) the ior at the surface of the dielectric sphere is defined by a noise-based texture that changes the directions of the internal and external transmitted rays. The result is an interesting image that looks as if it could have been done with bump mapping. The black areas near the edge of the sphere are caused by total internal reflection. There is much to explore with this technique.



(a) (b)

Figure 27. *(a) A dielectric sphere and a reflective sphere ray traced with max_depth = 3, (b) the scene in part (a) rendered with a textured surface ior on the dielectric sphere.*

5.2.2 Volume Variation

5.2.2.1 Generalized Luneberg Lenses

These are spheres, and were ray traced by Suffern and Getto (1991). The ior is

$$\eta(r) = (C - r^2 / R^2)^{1/2}, \qquad \textbf{Equation 16}$$

where R is the radius, and the constant C satisfies $C \geq 2.0$. The original Luneberg lenses have $C = 2.0$ so that η at the surface is 1.0. These have the property that any set of parallel rays that intersect the lens, exit it from a single point. See Luneberg (1964), and Figure 28, which is from Suffern and Getto (1991).

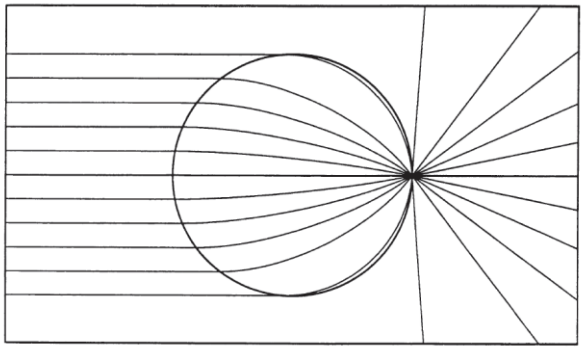
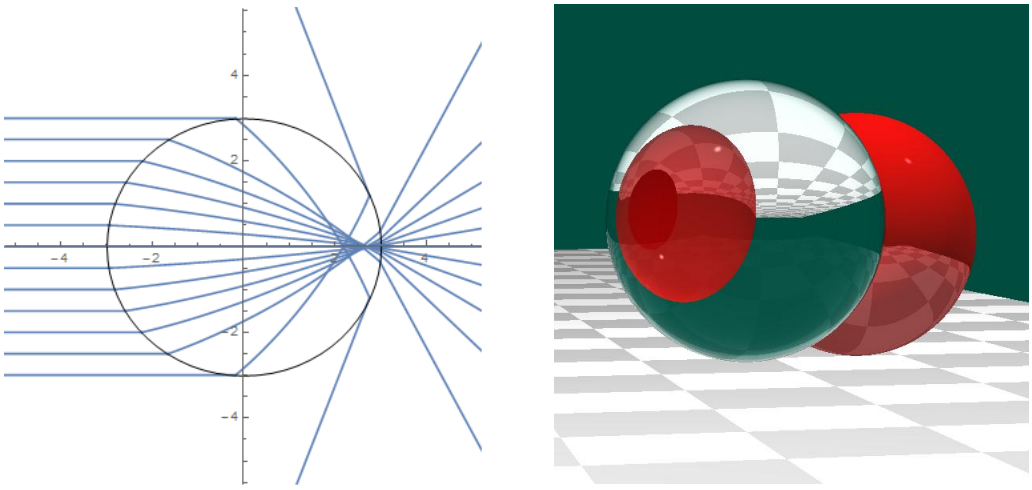


Figure 28. *Parallel rays that intersect a Luneberg lens all exit at a single point.*

Although there's a formula for the ray paths inside Luneberg lenses (Evans and Rosenquist, 1985), we have used numerical integration with the Runge-Kutta technique.

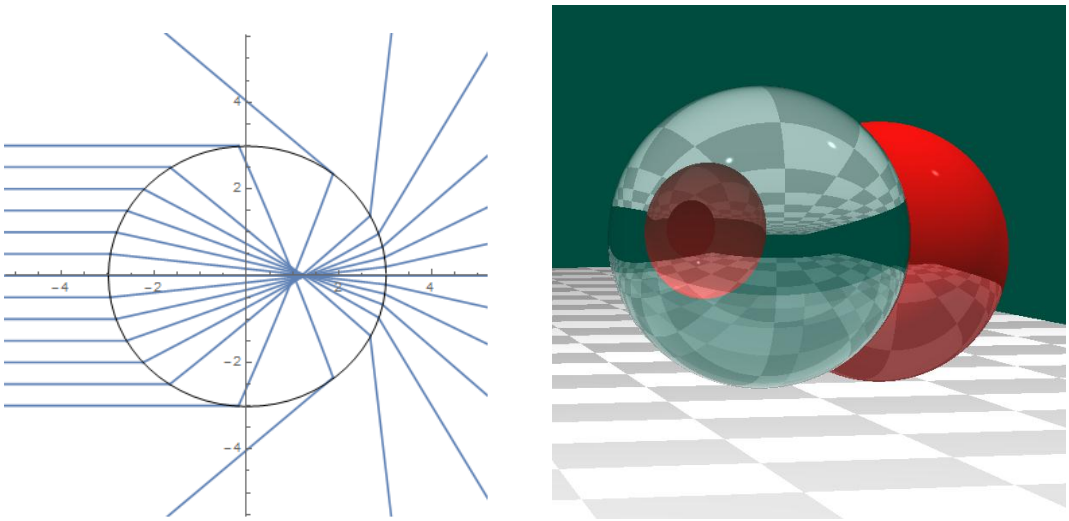
This is simpler, because it avoids a lot of mathematical analysis. Figure 29(a) shows a set of parallel rays hitting a sphere with C = 3.0. This sphere is ray traced in Figure 29(b).



(a) (b)

Figure 29. *(a) Internal and external rays for a Luneberg sphere with* $C = 3.0$ *and* $r_o = 3.0$*, (b) ray traced scene with this sphere.*

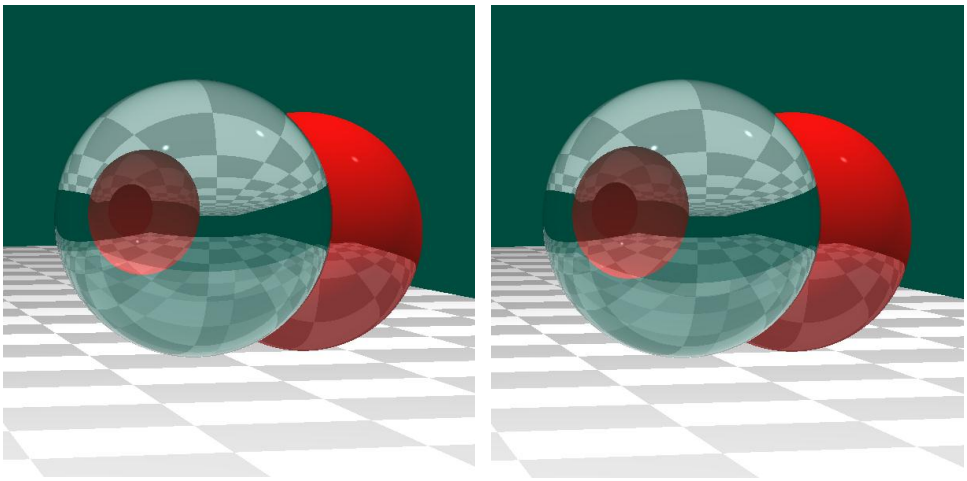
Figure 30 shows the rays and the ray traced image for C = 10.0. Here, the sphere is much more reflective than the one in Figure 29(b), because of the Fresnel equations. When C = 3.0, $\eta = \sqrt{2}$ at the surface, and at normal incidence $k_r = 0.029$ according to Equations (5) and (6). As a result, the reflection of the checkered rectangle is barely visible. In contrast, when C = 10.0, η = 3.0 at the surface, and at normal incidence k_r = 0.25, which is almost an order of magnitude higher. In Figure 30(b), the reflection is much brighter, and the refracted image of the rectangle at the top is not as bright.



(a) (b)

Figure 30. *(a) Internal and external rays for a Luneberg sphere with* C *= 10.0 and* R *= 3.0, (b) ray traced scene with this sphere.*

As *C* increases, the ray traced images become more like those with constant η. For example, the dielectric sphere in Figure 31(a) has η = 3.0 (which is not a physical material), and looks almost identical to the Luneberg sphere in Figure 31(b), which we have reproduced next to it make the comparison easier. The reason is that from Equation (16), as *C* increases, the difference between η at the center and the surface becomes smaller, and therefore more like a dielectric. In this example with *C* = 10.0, at the center $\eta = \sqrt{10.0} = 3.16$, and at the surface η = 3.0.



(a) (b)

Figure 31. *(a) Dielectric sphere with η = 3.0, (b) reproduction of Figure 6.6(b).*

5.2.2.2 Inverse r Spheres

(a) Analysis

In these spheres η is given by the simple expression

$$\eta = k / r \qquad \textbf{Equation 17}$$

where k is a constant, and r is the distance from the center. In contrast to the Luneberg spheres, where η is always finite, here η is infinite at the center. As such, these spheres are non-physical, but as we will see, the $1 / r$ behavior of η gives rise to a most interesting set of optical effects. These spheres were first studied by Evans and Rosenquist (1985) who found that the ray paths are *Logarithmic spirals*

$$r = ce^{dq} \qquad \textbf{Equation 18}$$

where c and d are constants. See Figure 32. As we'll see, the analysis is long and complex, but unavoidable. This because in theory, the light paths can have can have infinite length, and therefore we cannot use numerical integration. If we could, it would not give us the insight into the images that the analysis does.

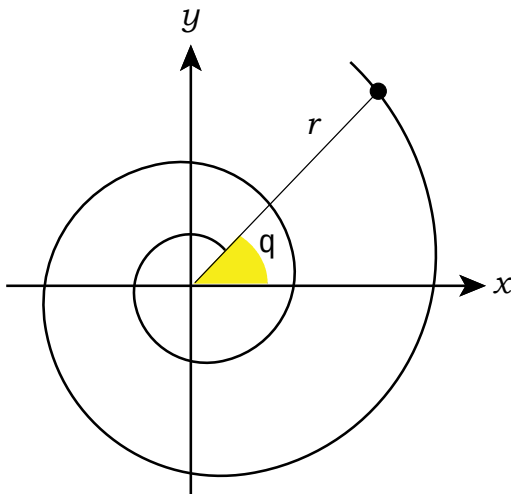


Figure 32. *A logarithmic (or exponential) spiral.*

When $d = 0$, these are circles of radius c. Evans and Rosenquist (1985) found that the time for light to traverse these circular orbits is independent of the radius.

Although the expression for the spirals in Equation (18) is simple, the following analysis for ray tracing purposes is surprisingly long and complex.

To ray trace inverse *r* spheres we put the camera inside the spheres because this produces the most interesting images. Each camera (primary) ray is a spiral in its own plane that passes through the sphere center $\boldsymbol{c} = (c_x, c_y, c_z)$, and is uniquely defined by the camera's eye point $\boldsymbol{e} = (e_x, e_y, e_z)$, and the initial unit ray direction $\boldsymbol{d} = (d_x, d_y, d_z)$. These are illustrated in Figure 33, where their (x, y, z) components are specified in the world coordinates (x_w, y_w, z_w) shown at the bottom left. Although the spiral is drawn from the center of the sphere, the only part that is relevant here is between $\boldsymbol{e}$, and the hit point $\boldsymbol{p}$ on the inside surface of the sphere. That is because in this case, $\boldsymbol{d}$ points away from the center. More generally, $\boldsymbol{d}$ can point in any direction, including towards the center, in which case the part of the spiral that goes to the center would be relevant. We'll discuss this in more detail below.

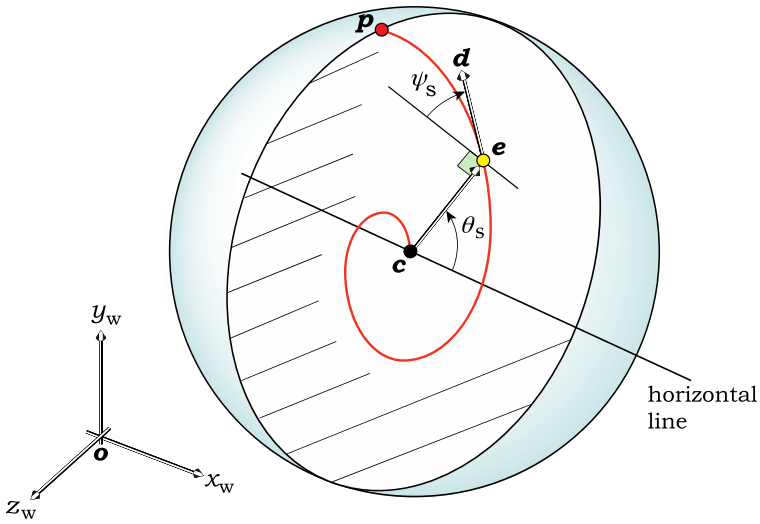


Figure 33. *Spiral configuration in an inverse sphere. The white ellipsoid is the circular disk that contains the spiral. It's defined by* **c**, **e**, *and* **d**, *and has radius* R *which is the sphere radius. The horizontal line is parallel to the* (x_w, z_w) *plane; the angle* ψ_s *is used to define the spiral through its initial direction.*

To trace a primary ray, we have to compute where the spiral hits the sphere, and the angle of incidence at the hit point. We can then use standard dielectric surface physics to test for total internal reflection at the hit point, and if there is none, compute the directions of the internal reflected ray, and the external transmitted ray, and trace them. We trace the external ray in the normal manner, but the internal ray is another spiral that starts on the surface.

To derive an expression for the spiral in Figure 33 we need to perform a number of steps. The first is to set up a 2D orthonormal frame with origin $\boldsymbol{e}$, in the plane as

shown in Figure 34. This consists of the two perpendicular unit vectors, $\boldsymbol{b}$ and $\boldsymbol{b}^{\perp}$, which meet at $\boldsymbol{e}$.

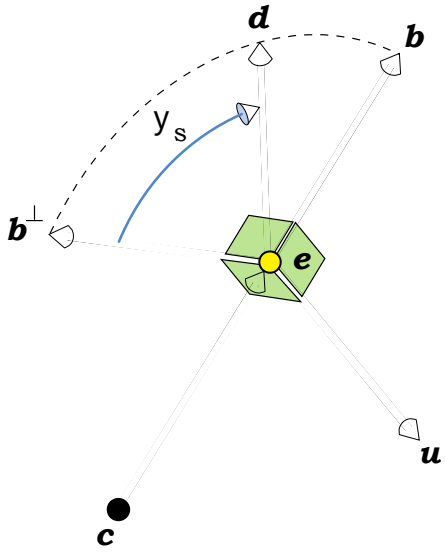


Figure 34. *Unit vectors used for calculating the spiral. The green squares indicate vectors that are perpendicular.*

We construct these as follows. First,

$$\boldsymbol{b} = (\boldsymbol{e} - \boldsymbol{c}) / \|\boldsymbol{e} - \boldsymbol{c}\|.$$

Next, we construct the unit vector $\boldsymbol{u}$, which is perpendicular to $\boldsymbol{b}$, $\boldsymbol{b}^{\perp}$, and the plane:

$$\boldsymbol{u} = (\boldsymbol{b} \times \boldsymbol{d}) / \|\boldsymbol{b} \times \boldsymbol{d}\|.$$

Finally,

$$\boldsymbol{b}^{\perp} = \boldsymbol{u} \times \boldsymbol{b},$$

which is a unit vector by construction. The unit vectors to $\boldsymbol{b}$, $\boldsymbol{b}^{\perp}$, and $\boldsymbol{u}$ are a right handed system, but $\boldsymbol{u}$ is only used to construct $\boldsymbol{b}^{\perp}$.

In our formulation of the spirals we use the angle ψ_s between $\boldsymbol{b}^{\perp}$ and $\boldsymbol{d}$, as shown in Figure 34. The subscript s indicates the start of the spiral at $\boldsymbol{e}$. Figure 35 shows a number of quantities that we need to compute for the spiral. This figure is a perpendicular view of the disk shown in Figure 33, which contains the spiral.

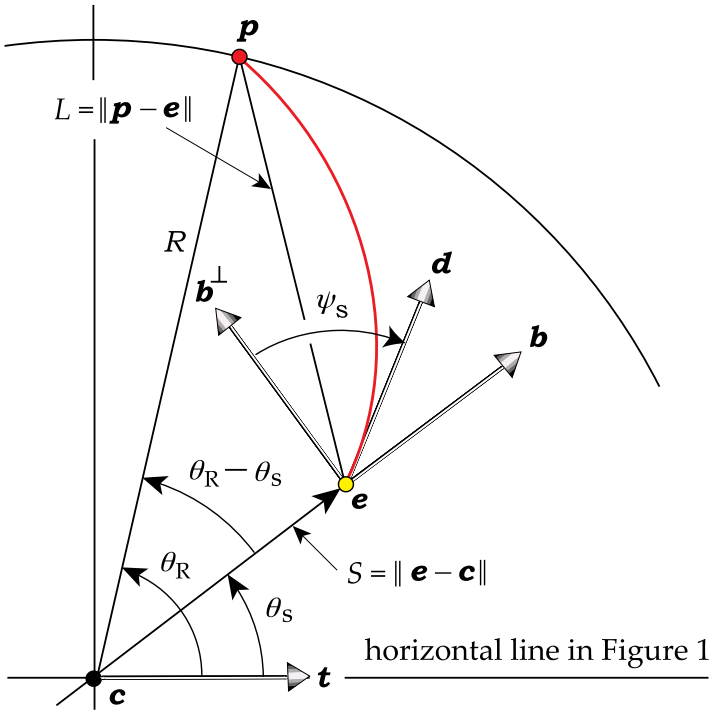


Figure 35. *Quantities that we need to compute the equation of the spiral, and the point* **p** *where it intersects the sphere surface.*

The equation of the spiral is

$$r = \| \boldsymbol{e} - \boldsymbol{c} \| \, e^{\tan(\psi_S)(\theta - \theta_S)} \qquad \textbf{Equation 19}$$

where $r = (x^2 + y^2 + z^2)^{1/2}$.

To compute θ_S, we use the dot product between the vectors $\boldsymbol{e} - \boldsymbol{c}$ and $\boldsymbol{t}$, as also illustrated in Figure 35. The expression for $\boldsymbol{t}$ is

$$\boldsymbol{t} = b_x^{\perp} \boldsymbol{i} + b_z^{\perp} \boldsymbol{k},$$

where $\boldsymbol{i}$ = (1.0, 0.0, 0.0) and $\boldsymbol{k}$ = (0.0, 0.0, 1.0) are the unit basis vectors in the x_w and y_w directions respectively. We then have

$$(\boldsymbol{e} - \boldsymbol{c}) \bullet \boldsymbol{t} = \| \boldsymbol{e} - \boldsymbol{c} \| \, \| \boldsymbol{t} \| \cos \theta_S,$$

so that

$$\theta_S = cos^{-1} \frac{(e_x - c_x)\, b_x^{\perp} + (e_z - c_z)\, b_z^{\perp}}{\| \boldsymbol{e} - \boldsymbol{c} \| [(b_x^{\perp})^2 + (b_z^{\perp})^2]^{1/2}}. \qquad \text{Equation 20}$$

The spiral hits the inside of the sphere where $r = R$ and $\theta = \theta_R$, as also shown in Figure 35. From Equation 19 we have

$$R = \| \boldsymbol{e} - \boldsymbol{c} \| e^{\tan(\psi_s)(\theta_R - \theta_s)},$$

so that

$$\theta_R = \theta_s + \ln[R / \| \boldsymbol{e} - \boldsymbol{c} \|] / \tan(\psi_s). \qquad \text{Equation 21}$$

We now have to compute the 3D coordinates $\boldsymbol{p}$ of the hit point, which is more complicated than computing θ_R. To do this we use the lemon triangle in Figure 36.

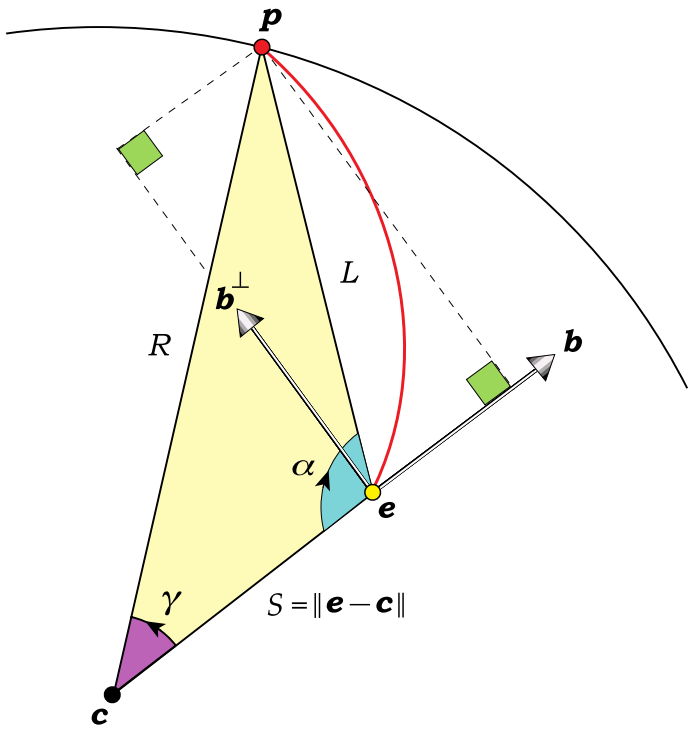


Figure 36. *The triangle we use to compute* $\mathbf{p}$.

Here, we know the two sides R and S, and that the angle γ is equal to $\theta_R - \theta_s$ from Figure 35. Using these quantities, we can use the cosine rule for triangles to derive the following expression for the third side L:

$$L = (R^2 + S^2 - 2RS\cos\gamma)^{1/2},$$

which is the distance between $\boldsymbol{e}$ and $\boldsymbol{p}$. The quantity we don't know is the angle α, but since we know the three sides L, R, and S and the angle γ, we can use the cosine rule again with these, to get the following expression for $\cos\alpha$:

$$\cos\alpha = (S^2 - RS\cos\gamma) / (LS).$$

Knowing L and α allows us to project L onto the $\boldsymbol{b}$ and $\boldsymbol{b}^{\perp}$ directions as shown in Figure 36. The expression for $\boldsymbol{p}$ is then

$$\boldsymbol{p} = \mathbf{e} - L\cos\alpha\ \mathbf{b} + L\sin\alpha\ \mathbf{b}^{\perp}. \qquad \textbf{Equation 22}$$

The minus sign in front of $L\cos\alpha\,\boldsymbol{b}$ is necessary because α is measured clockwise from S. In Figure 36, L is drawn to the right of the $\boldsymbol{b}^{\perp}$ direction, so that $\alpha < 90^{o}$ and $\cos\alpha < 0.0$. The minus sign makes the $\boldsymbol{b}$ projection positive in this case. If L was drawn to the left of the $\boldsymbol{b}^{\perp}$ direction, we would have $\alpha < 90^{o}$ and $\cos\alpha > 0.0$ to give a negative $\boldsymbol{b}$ projection, as required. Of course, the expression (22) has to be correct for all camera locations $\boldsymbol{e}$ inside the sphere, and all 3D primary ray directions from $\boldsymbol{e}$, which it is.

The last thing we need to compute is the tangent vector $\boldsymbol{\omega}_o$ to the spiral at the hit point $\boldsymbol{p}$, as illustrated in Figure 37. As these points back along the spiral, according to ray tracing convention, it is inside the sphere.

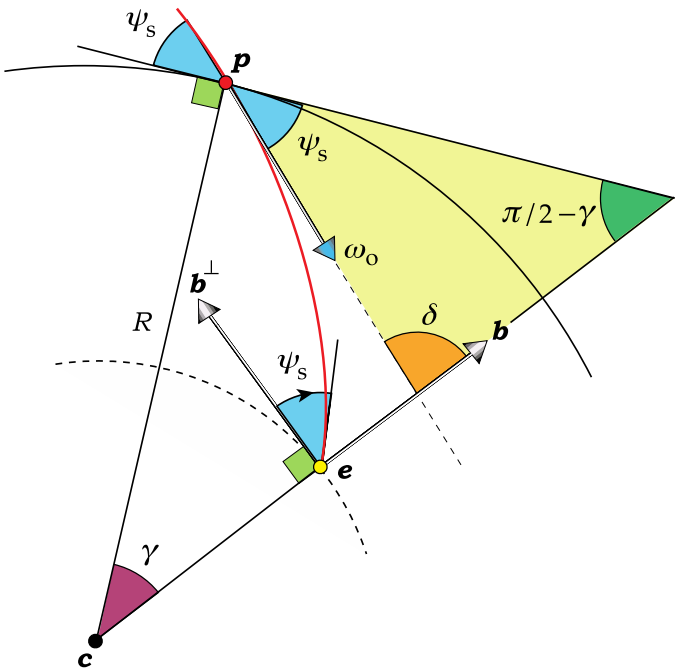


Figure 37. *The quantities we need to compute the tangent vector* $\boldsymbol{\omega}_o$ *to the spiral at* $\mathbf{p}$.

To compute $\boldsymbol{\omega}_o$, we use the fact that a logarithmic spiral makes the same angle to any circle that is centered on the origin, which is $\boldsymbol{c}$ in our case. Figure 37 shows parts of two such circles. The first goes through $\boldsymbol{e}$, where the angle is the specified value of ψ_s. The second is the surface of the sphere, where the angle of the yellow triangle at $\boldsymbol{p}$ is also ψ_s. We now need to project $\boldsymbol{\omega}_o$ onto the $\boldsymbol{b}$ and $\boldsymbol{b}^\perp$ directions by computing the angle δ. Fortunately, this is simple to do because

$$\gamma = \cos^{-1}[(\boldsymbol{p} - \boldsymbol{c}) \bullet \boldsymbol{b} / R],$$

and

$$\delta = \pi / 2 + \gamma - \psi_s.$$

Projecting $\boldsymbol{\omega}_o$ onto the $\boldsymbol{b}$ and $\boldsymbol{b}^\perp$ then gives

$$\boldsymbol{\omega}_o = -\cos \delta \, \boldsymbol{b} - \sin \delta \, \boldsymbol{b}^\perp,$$

which can be simplified to

$$\boldsymbol{\omega}_o = \sin (\gamma - \psi_s) \, \boldsymbol{b} - \cos (\gamma - \psi_s) \, \boldsymbol{b}^\perp,$$

where by construction, $\boldsymbol{\omega}_o$ is a unit vector.

To understand what happens to the spirals, we need to look at Figures 38 and 39. Figure 38 has a lot of detail, because it shows eight spirals in the same plane that leave $\boldsymbol{e}$ in different directions. The first thing to note is that there are two dashed lines that go through $\boldsymbol{e}$ and are parallel to $\boldsymbol{b}$ and $\boldsymbol{b}^\perp$. These divide the disk inside the sphere into four quadrants labeled I, II, III, and IV.

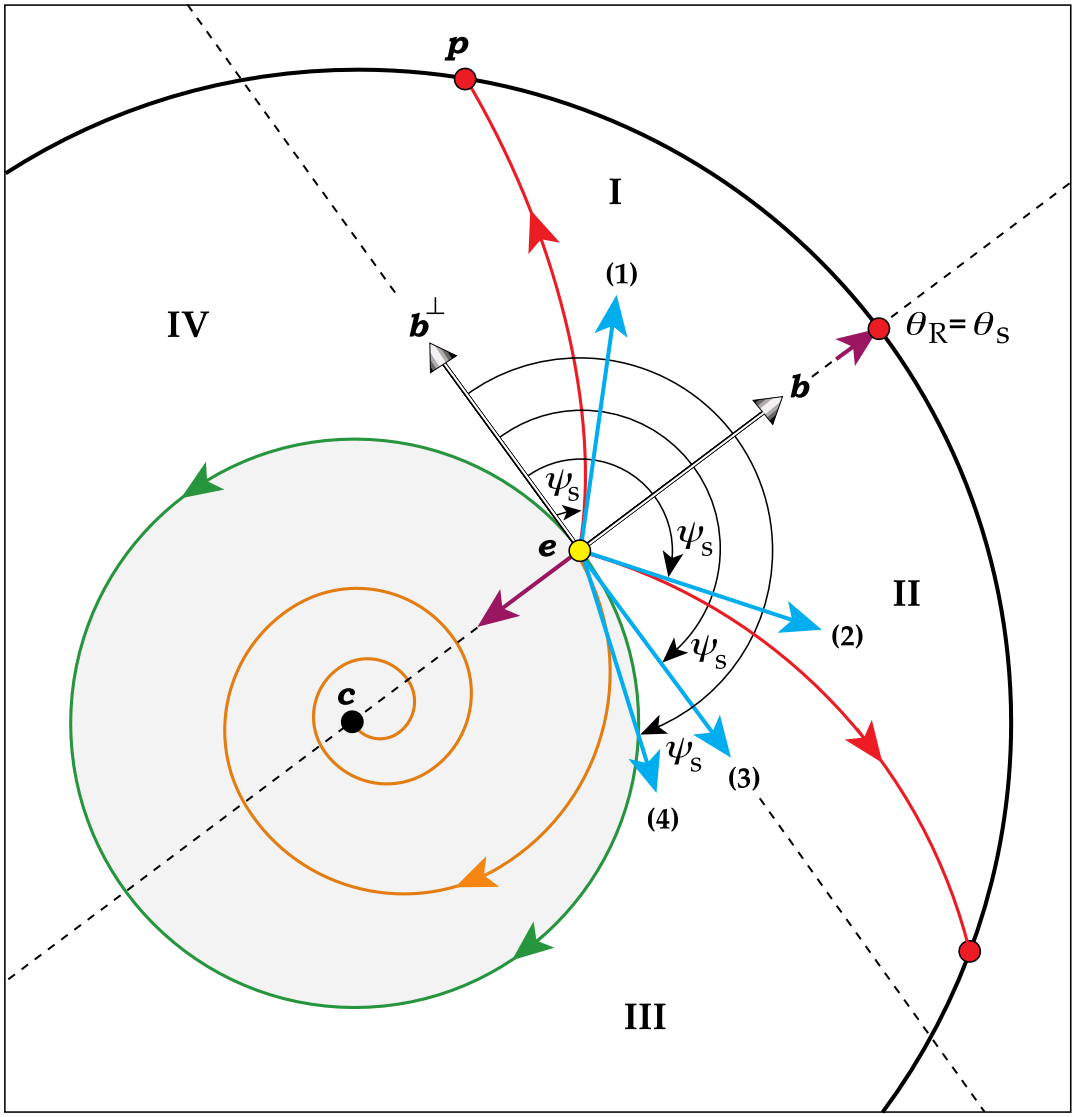


Figure 38. *Eight spirals in the same plane that start at* **e**, *and go in different directions. Two of the spirals are circles, and two are straight lines. This figure also shows how the two perpendicular (dashed) straight lines through* **e**, *and parallel to* **b** *and* $\mathbf{b}^{\perp}$, *divide the disk inside the sphere into four quadrants I, II, III, and IV.*

The quadrant that a spiral starts in, determines where it goes, but we'll mention a special case first. When $\psi_S = \pi / 2$, the spiral is parallel to $\boldsymbol{b}$, and is therefore a radial straight line as indicated by the purple arrows. This hits the sphere surface with $\theta_R = \theta_S$.

The four cyan arrows (1) – (4) are unit tangent vectors at the start of spirals. The spiral associated with the arrow (1), travels outwards in a counter-clockwise direction, until it hits the sphere surface. This is the red curve the hits the surface at $\boldsymbol{p}$. All the spirals with $\psi_S \in (0.0, \pi / 2)$ have this behavior. But from equation (21), $\theta_R \to +\infty$ when $\psi_S \to 0.0$, because the variable term involves $1.0 / \tan(\psi_S) = \cot(\psi_S)$. This is

illustrated in Figure 39, which is a plot of $\theta_S + \cot(\psi_S)$ for $\psi_S \in (0.0, 2\pi)$. Here, we have replaced the $\ln[R/\|\boldsymbol{e}-\boldsymbol{c}\|]$ term in Equation (21) with 1.0, because R, $\boldsymbol{e}$, and $\boldsymbol{c}$ are arbitrary, and this is just a vertical scaling factor. This plot therefore just shows the general shape of the θ_R curve.

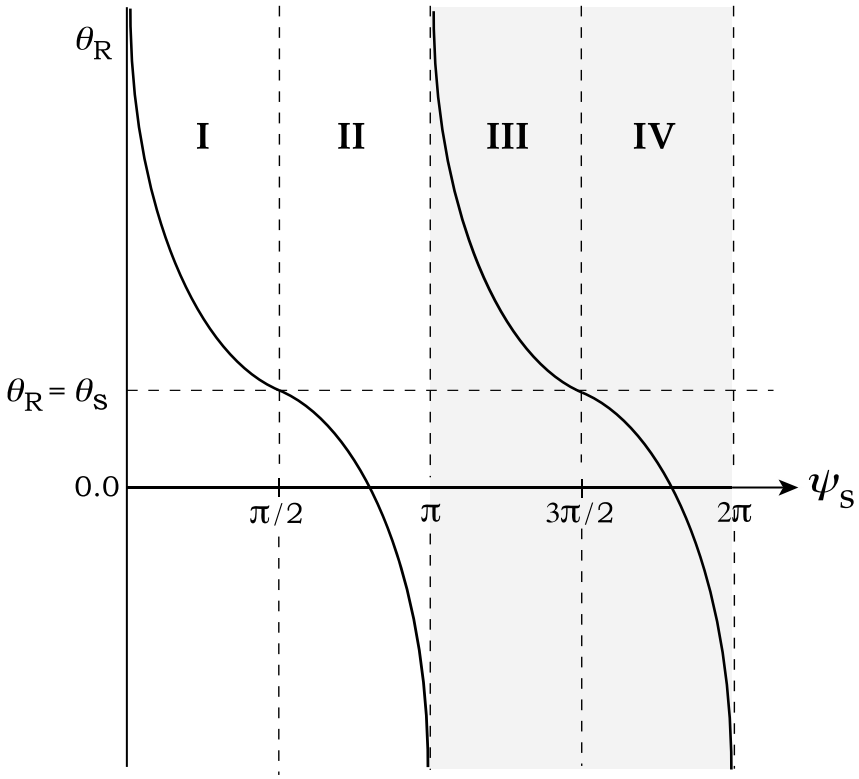
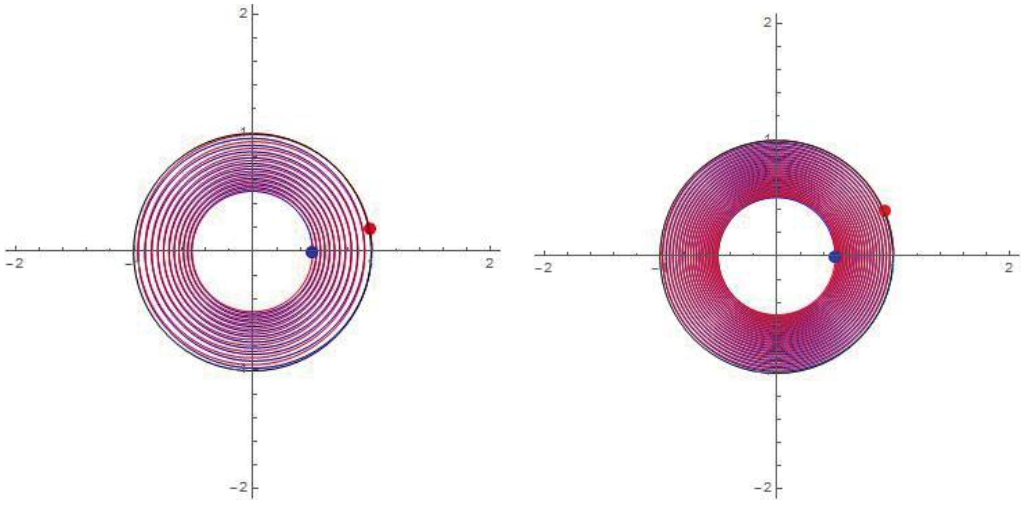


Figure 39. *Plot of* θ_R *as a function of* ψ_S.

As $\psi_S \to 0.0$ the spirals become more tightly wound and circular in shape, and their lengths increase without limit. This is why we cannot use numerical integration for the ray tracing, which would have saved us from a lot of the above mathematical analysis, but would have also provided no insights into what happens to rays inside the spheres. Figure 40 shows Mathematica plots of spirals with ψ_S = 0.1 and 0.01. We can see the effects of these tightly wound spirals in the ray traced images below.



(a) (b)

Figure 40. *Plots of tightly wound spirals with ψ_S = 0.1 in (a) and ψ_S = 0.01 in (b).*

When ψ_S = 0.0, the spiral is the green counter-clockwise circle on the edge of the gray disk in Figure 38. It is extremely unlikely that this will occur in ray tracing because we use multi-jittered sampling for antialiasing, where the rays start at random points on the surface of each pixel.

In quadrant II the spirals travel outwards, but in a clockwise direction. The spiral (2) is an example. As $\psi_S \rightarrow \pi / 2$, the spirals behave the same as they do when $\psi_S \rightarrow$ 0.0 except that they travel in a clockwise direction. When $\psi_S = \pi / 2$, the spiral is the green clockwise circle on the edge of the gray disk.

When ψ_S is in quadrant III with $\psi_S \in (\pi, 3\pi/2)$, the spirals go in to the center as illustrated by the clockwise orange spiral that starts in the direction (4). As $\psi_S \rightarrow \pi/2$ from the counter-clockwise direction, (that is $\psi_S \downarrow \pi/2$), the inward going spirals in the gray disk become more circular, in the same way as they do in quadrants I and II. When $\psi_S = 3\pi/2$ the spiral is a straight line into the center as indicated by the purple arrow in the disk.

When ψ_S is in quadrant IV, the spirals are the same as in quadrant III, except that they go in a counter-clockwise direction. We have not drawn any of these in Figure 38 with $\psi_S \in (3\pi/2, 2\pi)$.

Because the camera rays in quadrants III and IV go to the center where they are scattered, they do not form an image. We have to assign black to these pixels, but that is

not strictly correct, as discussed below. Equation (21) for θ_R still gives values in these quadrants, as shown on the right side of Figure 39 with the gray background. This corresponds to the gray disk in Figure 38. The curve is exactly the same as it is in quadrants I and II, but the fate of the spirals is the same, regardless of how long they are. Although Figure 38 is drawn with $\boldsymbol{e}$ in a specific location, the analysis holds for all locations of $\boldsymbol{e}$ inside the sphere. In summary, only rays in quadrants I and II can get out of the sphere, and as rays approach the $\pm\boldsymbol{b}^{\perp}$ directions from either side, their lengths approach infinity.

What really happens to the rays that as they approach the center? The simple answer is, we do not know, but we can make an educated guess. It is likely that as the spiral approaches the center, the geometric optics abstraction we use for ray tracing, where light travels on infinitely thin rays, will break down. This would be caused by infinite value of η at the center. It is then likely that the light is scattered about the center, and that we would have to use the wave theory of light, as described by Maxwell's equations to compute what happens. See for example, Hecht (2001). This would be a complex process, and it is not worth doing because this is a made-up situation.

(b) Results

For the ray tracing, we use the scene shown in Figure 41, where the black sphere at the center is the inverse sphere. The remainder of the scene is a ring of colored spheres and a checker plane. There is also a blue background, which is not visible in this image.

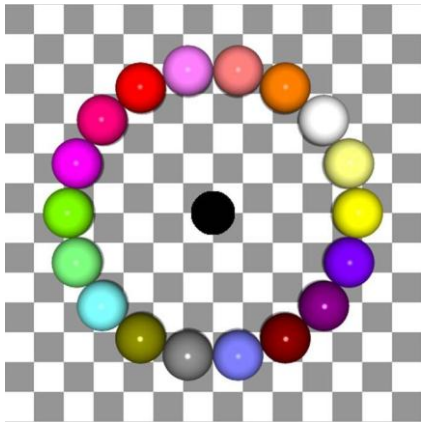


Figure 41. *The scene ray traced in the images below. The inverse sphere in the middle is centered on the world origin, and has radius 4.0. The* x_w *axis points to the right, the* y_w *axis points out of the paper, and the* z_w *axis points straight down.*

The inverse sphere is black because any rays that hit it from the outside generate spirals that end up at the center. Figure 42 shows a set of parallel rays that hit an inverse sphere from the outside, with $\eta = 1.0$ at the surface.

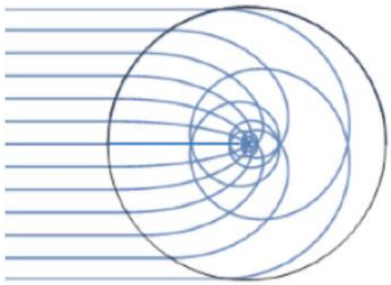
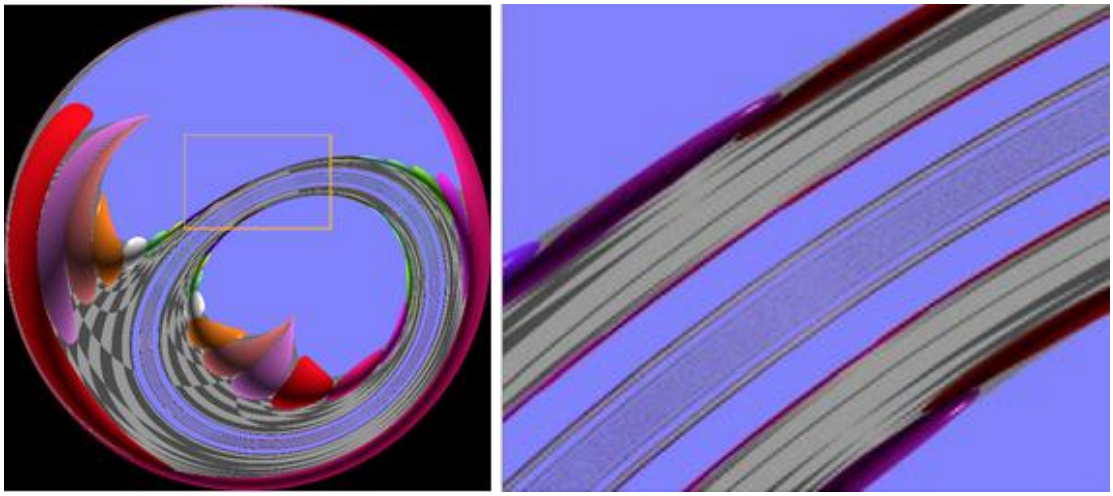


Figure 42. *A set of parallel rays entering an inverse sphere.*

For rendering scenes with the camera inside, a fisheye camera is best because it allows a wider field of view than a pinhole camera. The image in Figure 43(a) was rendered with a 180º fisheye camera, where the primary rays go in a hemisphere from the eye point, and centered on the viewing direction. Here, the image is the disk; the black areas in the corners are there because the fisheye camera produces square images. These could be any color, including white. The figure shows distorted images of the spheres, but as it turns out, the most interesting thing is the oval shaped band of what looks like noise. Figure 43(b) is a zoomed image of a part of the band that is in the rectangle in part (a). From the top left we have background, stretched spheres, the checker plane, and then even more stretched spheres. This pattern then repeats, and becomes smaller and smaller until it disappears into noise. Admittedly that's hard to see in this image. This is a characteristic of *fractals,* which are *self-similar*. This means that, no matter how far we zoom in, the central part of the image will always look similar, and the noise band will never be resolved. See Mandelbrot (1982).

The fractal is caused by the fact that $\theta_R \to +\infty$ when $\psi_s \to 0.0$, as discussed above. This is a nice discovery, and unexpected. But with hindsight (which is a wonderful tool), we could have predicted it from the above analysis and Figures 38 – 40. The images in Figure 47 will provide further evidence that the fractal is there.



(a) (b)

Figure 43. *(a) A 180^{O} fisheye camera image when the camera is inside the sphere, (b) a zoomed image from inside the rectangle in (a).*

The images in Figure 43 are technically incorrect because half of the noise band, and the area inside it, are where the camera rays start in the quadrants III and IV in Figure 38. Because these go into the center of the sphere, these pixels should be black, but here we have continued to use the formula for θ_R in Equation (21), and traced the rays. The graph for θ_R in these quadrants is in the right half of Figure 39 with the gray background. The graph is identical to one in left half, and therefore the fractal pattern on the inner half of the noise band is similar to the one on the outside.

Figure 44 is Figure 43(b) rendered with black pixels. The border is in the middle of the noise band, where $\theta_R = \pm\infty$. In the following images, we have used both techniques. Although there's no technical justification for tracing all the rays, it does make the images more interesting.

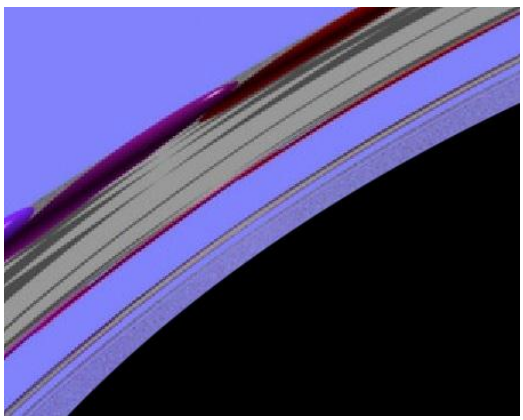
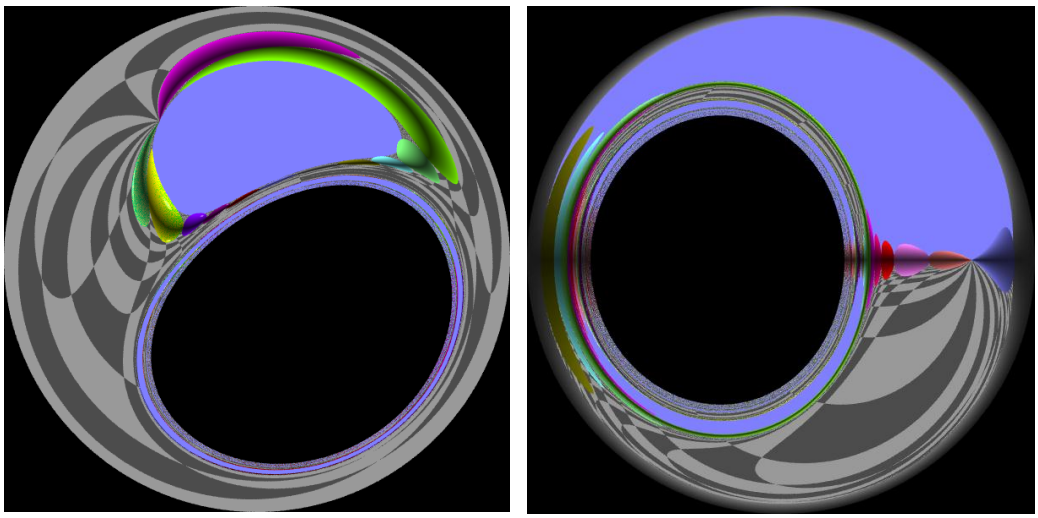


Figure 44. *Figure 43 rendered with black pixels for the in-going rays.*

Figure 45 shows two more 180º fisheye camera views with only the outgoing rays rendered. In part (b) the camera is looking horizontally.



(a) (b)

Figure 45. *Two 180º fisheye camera images where the camera is inside the sphere.*

The best way to demonstrate that a fractal is present is to repeatedly zoom into an image, and observe what happens. A fisheye camera is also best for this because we can zoom in by reducing the field of view (fov). Figure 46 shows fisheye images where the camera is looking horizontally, and at right angles to the radial direction. This is in the $\boldsymbol{b}^{\perp}$ direction in Figure 6.14. We have also traced all the rays. The images start with fov = 360º in (a), and end with fov = 2º in (f). These figures show progressively smaller amounts of the background color because two of the spheres touch in the middle of the images. Parts (e) and (f) look like abstract art, but need to be seen in high resolution square images to be fully appreciated. Unfortunately, the shading of the external spheres is not consistent in these images. As we zoom in, the details on either side of the noise band changes, but the noise band is always there. This is similar to zooming into the Mandelbrot set (Mandelbrot, 1982), which is two-dimensional, but here the fractal near the horizon is one-dimensional. Part (a) shows that the noise band splits in two at the top and bottom of the image, but the only way to see that in more detail would be to render higher resolution images.

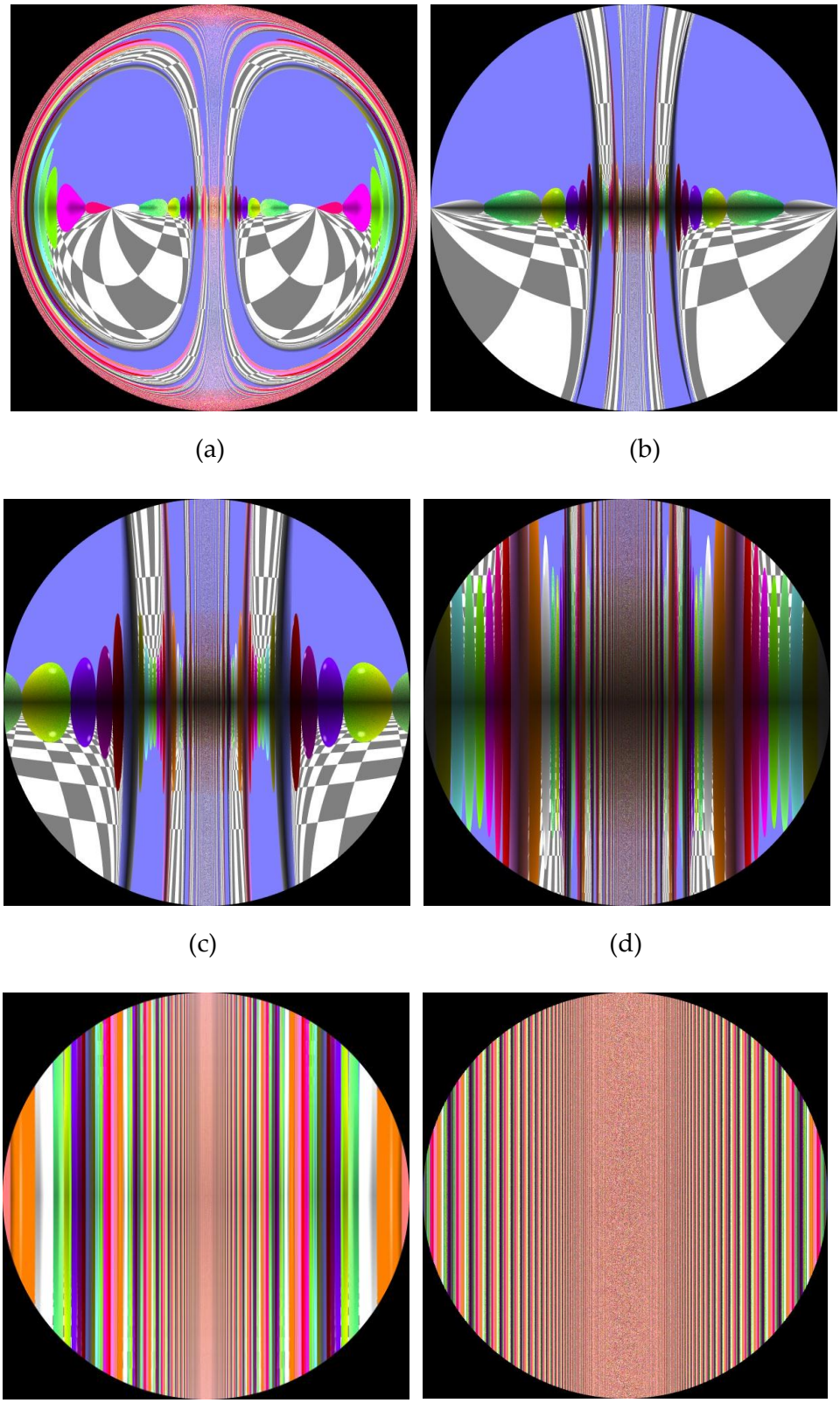
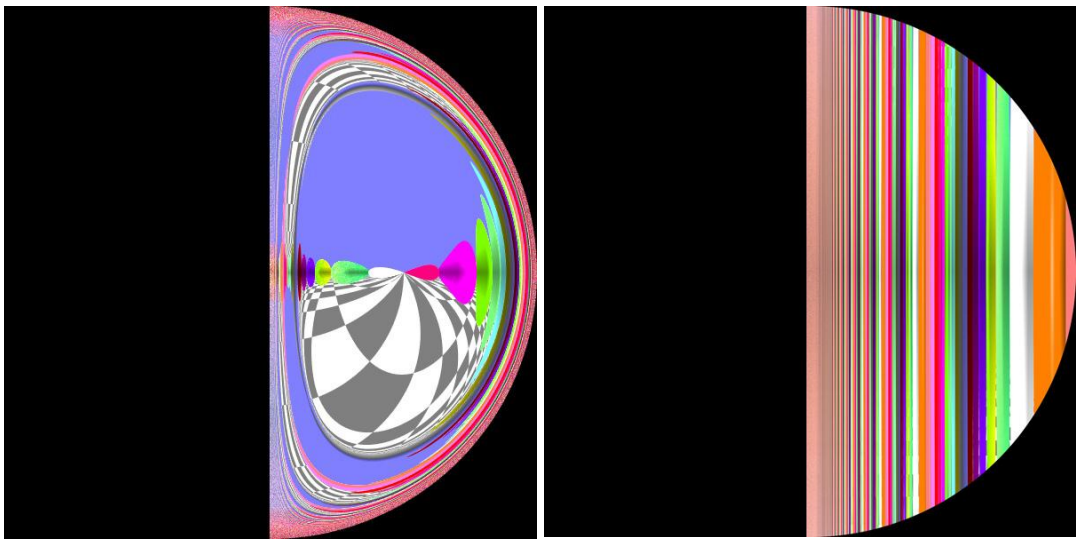


Figure 46. *Horizontal fisheye camera images with fov = 360^{O} in (a), 180^{O} in (b), 90^{O} in (c), 32^{O} in (d), 16^{O} in (e), and 2^{O} in (f). The camera is located at (2.0, 0.0, 0.0), and the look-at point is (2.0, 0.0, -2.0).*

Because the noise band is much wider in Figure 46(f), than it is in (e), it may appear that the fractal is not self-similar, but this is caused by the low effective pixel resolutions of the displayed images, which cannot resolve the fine details near the centers. Figure 47 shows expanded strips across the centers of both images, which almost identical.

Figure 47. *Expanded strips cross the centers of Figure 46 parts (e) and (f), with (e) on the top.*

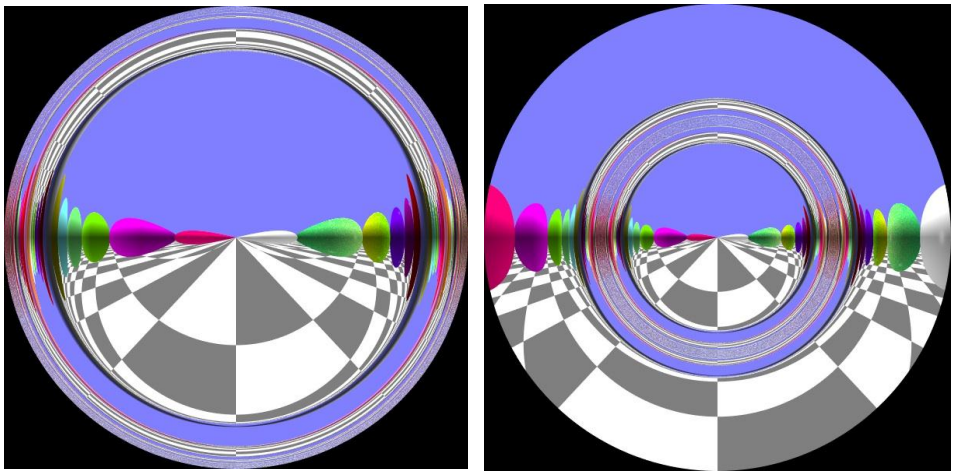
Because of the camera orientation in Figure 46, if we did not trace the ingoing rays, the left half of each image would be black. Figure 48 shows the results for parts (a) and (e).



(a) (b)

Figure 48. *Correctly rendered versions Figure 6.25(a) in (a) and 6.25(e) in (b).*

Figure 49 shows two views where the camera is looking radially out of the sphere in the x_w direction, with the look-at point (4.0, 0.0, 0.0). In part (a), where the fov = 180^o, the noise band is around the edge of the image. This is where $\psi_s \to 0.0$ and $\psi_s \to \pi / 2$ in quadrants I and II. Here, no primary rays are shot into quadrants III and IV. In part (b) where the fov = 360^o, the image in (a) is compressed towards the center, and there is no noise band around the edge.



(a) (b)

Figure 49. *Horizontal fisheye camera images looking radially out of the sphere with fov = 180^{O} in (a), and fov = 360^{O} in (b).*

Figure 50 shows the result when we return black for the rays in quadrants III and IV, which is correct, but doesn't have quite the appeal of Figure 49(b).

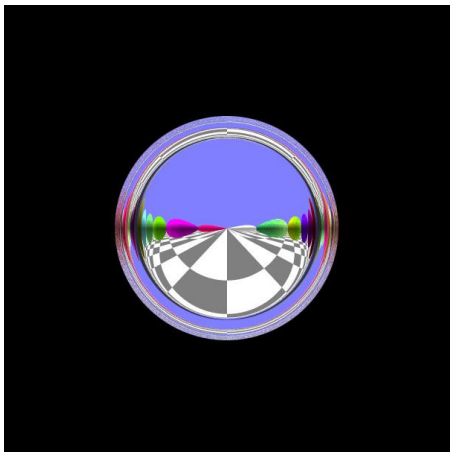


Figure 50. *Correctly rendered version of Figure 6.25(b).*

A nice feature of the Fisheye camera in the ray tracer is that we can use a field of view that is larger the 360º. This can create multiple wrap-around views of a scene. As an example, Figure 51 is Figure 49 rendered with the fov = 3600º. This is of no use scientifically, but it is interesting to look at. The small inner circle is the 360º view from Figure 46(a), but rendered with the dark spheres in parts (b) – (d). This is surrounded by 9 wrap-around images.

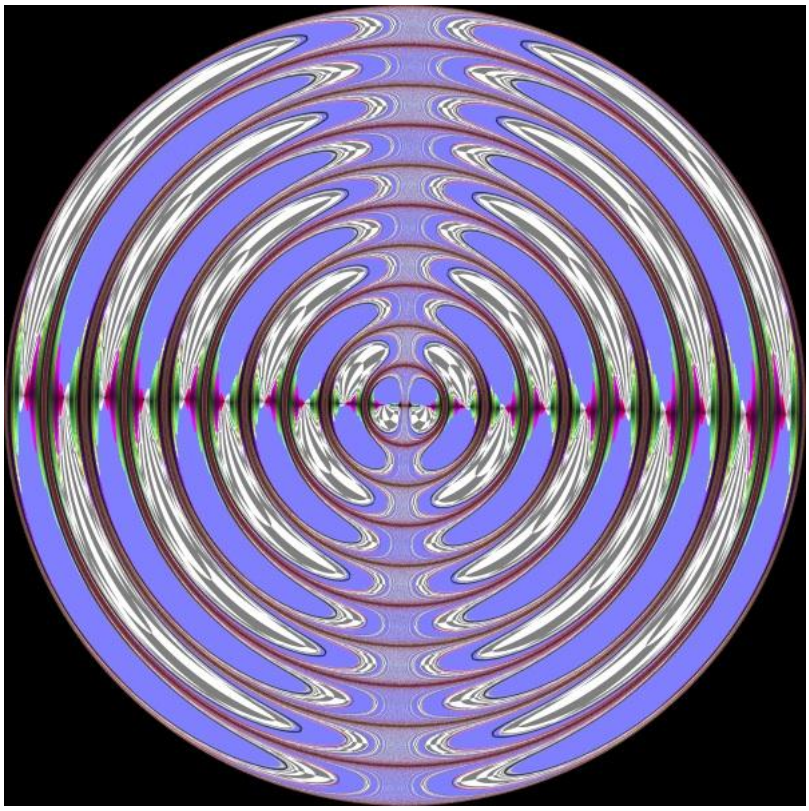
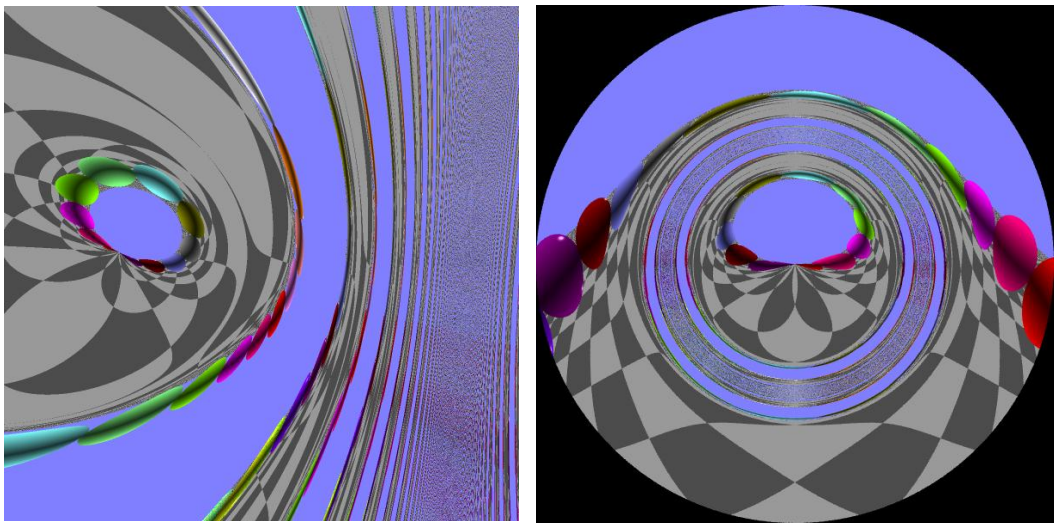


Figure 51. *Fantasy version of Figure 6.28(b) rendered with the fov = 3600°.*

Figure 52(a) is a pinhole camera image where striations are visible in the noise band. These are just the red spheres stretched out. The noise band in the fisheye image in part (b) appears to have a specular highlight at the bottom, but this is an illusion created by the light gray checkers lined up underneath it. We would need a much higher resolution image to see this properly.



(a) (b)

Figure 52. *(a) Pinhole camera image, (b) fisheye camera image with fov = 360°.*

In Figure 53 the inverse sphere has been shifted vertically in the y_w direction, which allows distorted top-down images of the ring of spheres to be rendered. These look like the petals of a flower.

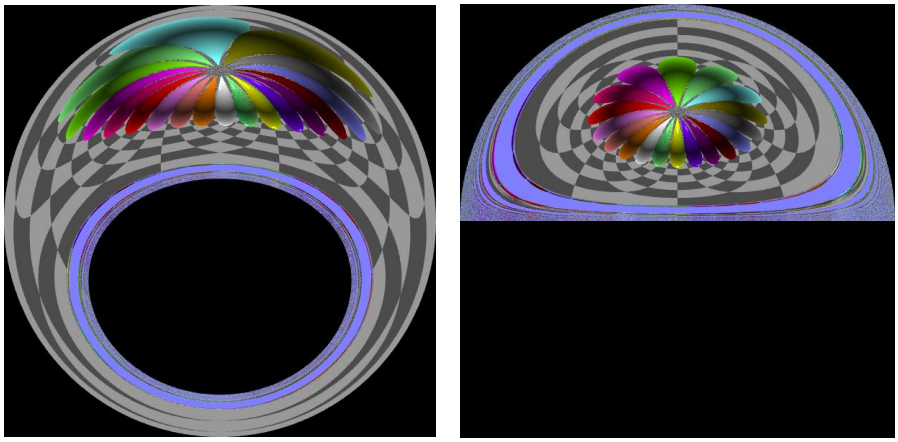


Figure 53. *Two fisheye camera images where the inverse sphere has been raised vertically.*

6 Realistic Ray Tracer

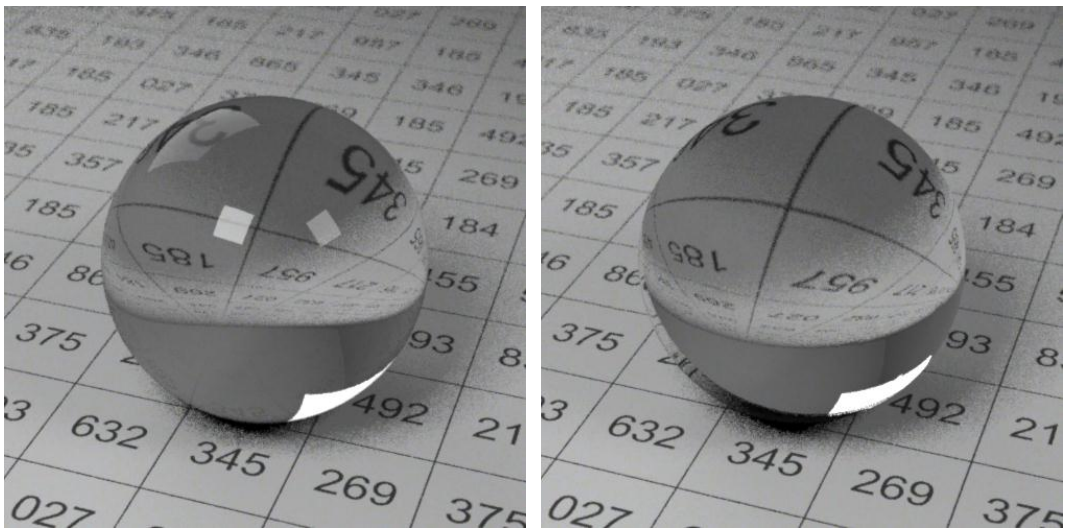
In the previous chapter we have used standard ray tracer and standard 3D objects to render and study different materials. e.g. glossy materials and variable (ior). In this chapter we have extended our work to work with more sophisticated software and more complex 3D models.

6.1 LuxRender

We ported our Runge-Kutta implementation to LuxRender, which is a multi-platform open source physical based renderer. See latest version www.luxcorerender.org . Using LuxRender allows us to render more realistic images of spheres with spatially varying indices of refraction because it is physically based. We have tested our Runge-Kutta technique with Luneberg and Gaussian spheres.

6.1.1 Generalized Luneberg lenses

Figure 54 shows two Luneberg lenses rendered with an area light. In part (a), the constant C in Equation (16) is equal to 3.0, and in part (b), $C = 2.0$. In (a) the reflection of the light is visible on the inside and outside surfaces of the sphere, because according to Equation (16), $\eta = \sqrt{2.0} = 1.414$ at the surface. In contrast, there are no reflections in (b) because when $C = 2.0$, $\eta = 1.0$ at the surface, and therefore $k_r = 1.0$ on the inside and outside surfaces according to the Fresnel Equations (5) – (7).



(a) (b)

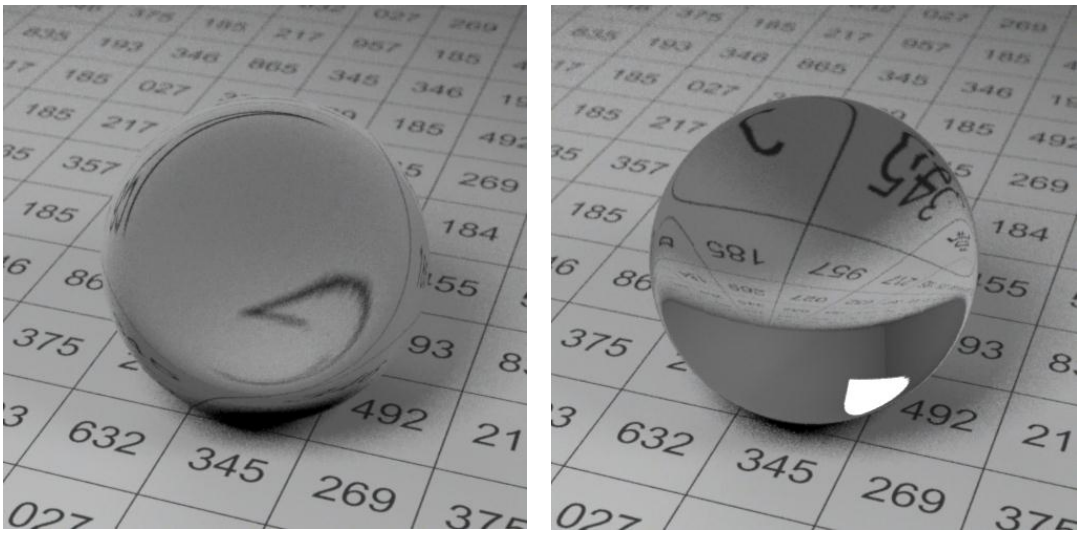
Figure 54. *Luneberg spheres rendered with* C *= 3.0 in (a) and* C *= 2.0 in (b).*

6.1.2 Gaussian spheres

In these spheres, η is given by the inverse exponential function

$$\eta = a\, e^{-b\frac{r^2}{R^2}}$$

where a and b are arbitrary constants, and R is the radius. This is also known as a Gaussian function. When $r = 0.0$, $\eta = a$, and when $r = R$, $\eta = a e^{-b}$. When $b = \log a$, $\eta = 1.0$ at the surface for all values of the radius. Figure 55 shows two Gaussian spheres rendered with $\eta = 1.0$ at the surface. In part (a), $a = 1.25$, and in part (b), $a = 1.4$. As we can see, the small change in a makes a big difference in the refraction.

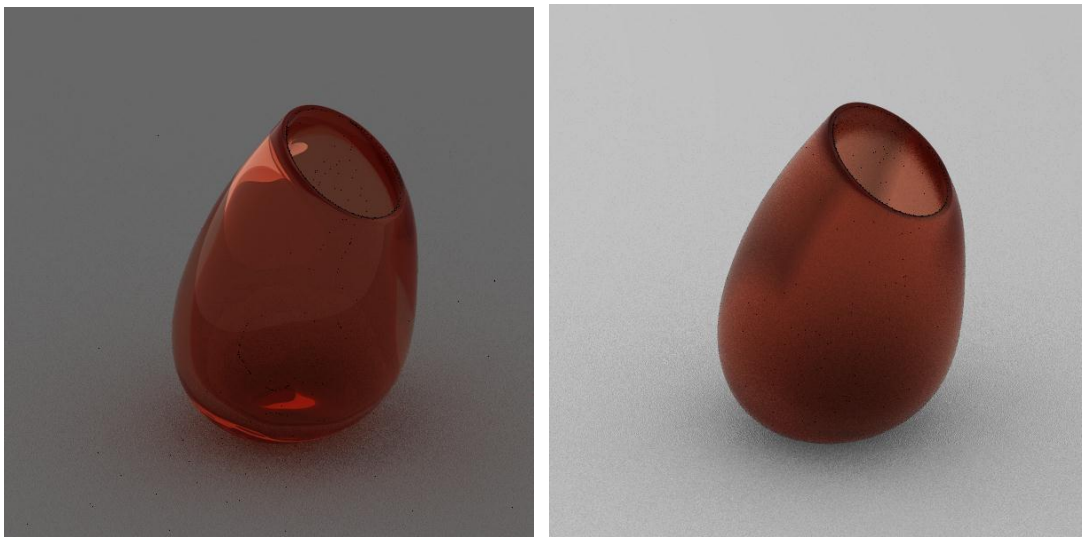


(a) (b)

Figure 55. *Gaussian spheres rendered with* a = *1.25 in (a) and* a = *1.5 in (b).*

6.2 Vase Case Study

We modeled a vase in Autodesk Maya with a high-density triangle mesh, and rendered it with different materials and rendering techniques. In Figure 56(a), the vase has a dielectric material with a constant filter color, and it has been rendered with an area light and path tracing. This has produced a colored caustic. In Figure 56(b) the material is a glossy transmitter with a noise based filter color, and is rendered with Whitted ray tracing and ambient occlusion to produce the soft shadow.



(a) (b)

Figure 56. *(a) Vase with dielectric material and constant color filter, (b) vase with glossy transmitter material and a noise-based filter color.*

As a final result, we rendered the vase with a glossy transmitter material, a rainbow ramp texture filter, an area light, and path tracing, to produce a colored caustic. The noise is present because we were not able to use enough samples per pixel. Combining glossy transmission, a noise-based textured filter color, and path tracing with an area light source that was just large enough to produce soft shadows, took a long time to render. But the result is still a nice image.

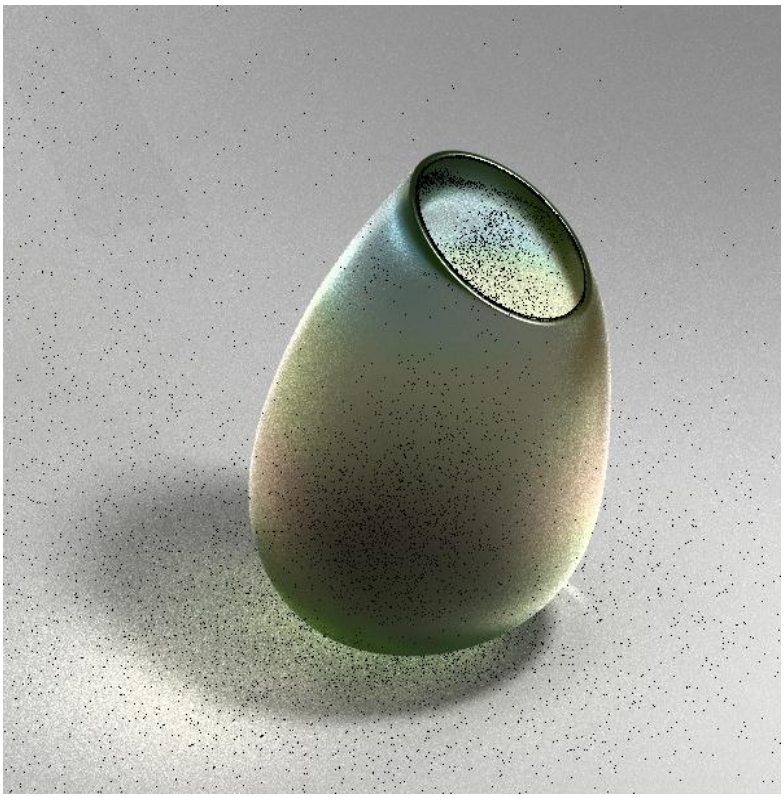


Figure 57. *The final image of the vase.*

7 Discussion

In this chapter we discuss the related issues we faced when applying features to the ray tracer.

7.1 Textured Filter Colors

We have used Whitted style ray tracing to render dielectric materials where the filter color is a noise-based 3D texture. To compute the total filter color along each ray, we sampled the texture at multiple points along the rays, and used Beer's law to compute the amount of filtering between each pair of points. Because the texture is the sum of a random function and its octaves, the points have to be close enough together to accurately sample the highest octave. We need to develop a test for this.

We also rendered a material with a wrapped noise-based texture where there are finite discontinuities in the texture, and therefore the color. These result in ridges in the rendered images. Previously, wrapped textures had only been rendered on opaque materials. If wrapped textures were used with glossy transmission, they may be able to simulate cracks in materials like frosted glass. We need to investigate this.

7.2 Surface Based Variable Index of Refraction

We rendered a clear dielectric sphere where the index of refraction η at the surface is determined by a noise-based texture, but inside η is constant along each ray. This resulted in an interesting image, but we need to explore this technique with different surface textures, and combine it with interior textured filter colors.

7.3 Interior Variable Index of Refraction

We ray traced three types of spheres where the index of refraction varies with the distance from their centers.

7.3.1 Generalized Luneberg Lenses

These were ray traced by Suffern and Getto (1991) using the analytic formula for the interior ray paths. In contrast, we used Runge-Kutta numerical integration because this is often simpler to implement. The formula for η is in Equation (6.1), where C is a constant. As C increases, η increases as well, but varies less from the center

to the surface. For large values of C, the spheres therefore become like constant η dielectrics, but with un-physically large values of η. We used Mathematica to plot a set of internal and external rays for C = 3.0 and C = 10.0, and ray traced a scene of these spheres with the camera outside. Because of the Fresnel equations (5) – (7), the spheres become more reflective as C increases. This is evident in Figure 30(b) where the reflections of the checkered square and backgrounds are much brighter than they are in Figure 29(b), where C = 3.0.

7.3.2 Inverse *r* spheres

Here, $\eta = k / r$, where r is the distance from the center, and k is a constant. This is as simple as it can get, and the ray paths are logarithmic spirals. But in spite of their simplicity, these spheres produced by far the most interesting results when we placed the cameras inside them. The reason is that circles are a special case of the spirals, and as the spirals become more circular, their lengths inside the spheres increase without limit. As an example, Figure 42 shows two nearly circular spirals. This had two consequences. First, it prevented us from using numerical integration to trace the rays, so that we had to use mathematics to compute where the camera rays went, and what happened to them. This process was surprisingly long and complex. Second, it produced fractals in the images which could not have been rendered with numerical integration. These are created as the camera ray directions approach a plane that's perpendicular to the line between the center of the sphere and the camera's eye point. Figure 38 illustrates these in 2D; Figure 39 illustrates how the angle θ_R, where the ray hits the inside of the sphere, approaches $\pm\infty$ as the ray directions approach the plane.

To show that the images really are fractals, we used a fisheye camera looking in the plane, and zoomed in from a 360º view to a 2º view. The resulting images are in Figures 46 and 47. In Figure 46(a) the fractal runs up and down the middle of the image, where it is hard to see, and then runs around the edge. Figure 47 shows that the fractal is self-similar, because the general appearance of the central part doesn't change as we zoom in. Figure 46(e) looks like an example of modern art, but it would look better if it was rendered with a pinhole camera, because that produces square images.

These images are not strictly correct because in their left halves, the rays go into the center of the sphere, and so these should be black. We have also rendered images where we have done this correctly.

There is more work to do with these spheres. The images we have rendered all have $\eta = 1.0$ at the surface, and therefore there is no refraction. We need to render these with $\eta_S > 1.0$. Although $\eta = 2.42$ for diamond is the largest η for real dielectrics (Table 1), there is no limit for η_S with the inverse spheres. We can therefore explore the optical environment inside the spheres with arbitrarily large values of η_S. There are however, surface effects that should be taken into account. According to the Fresnel equations (5) – (7), in the limit $\eta_S \to \infty$, a dielectric looks like a perfect mirror viewed from the outside, and no light gets in. When viewed from the inside, the surface also appears as a perfect mirror, and no light gets out. If we were ray tracing an inverse sphere from the inside, the outside view would therefore become darker as η_S increased. (Of course, the same surface behavior would apply any object with a constant η.)

Another problem we would like to look at involves placing a reflective sphere inside an inverse sphere. If we ray traced the inverse sphere from the outside, it would no longer appear black because some rays would hit the sphere, be reflected off it, leave the inverse sphere, and then return radiance to the camera. What would the reflective sphere look like? This would very much depend on its size, and where it is inside, relative to the camera. What would it look like when the camera is also inside the inverse sphere?

The mathematical problem can be stated quite simply. How do we intersect a logarithmic spiral with a circle? Unfortunately, the answer is not simple to find, because the spiral can intersect the circle an unlimited number of times as it becomes more circular, and we need to find the first intersection along the spiral from the camera. One approach would be to use the formula for the arc length along the spiral to compute a series of points along it, starting from the camera. If the distances between these were considerably smaller than the sphere radius, we could compute the square of the distance between each point and the sphere center, and stop when it's less than the square of the sphere radius. We could then use linear interpolation between the last two points to compute the intersection with the sphere. This should work provided we keep the sphere away from the fractal zones. We could then apply this technique recursively to the reflected rays, which could also hit the sphere, and could also be in a fractal zone

as seen from their starting points on the sphere. We would also have to test if the rays have hit the surface of the inverse sphere, and do this at every step. Finally, we would have to stop the process after some maximum number of points have been generated with no intersection. It would be fascinating to see what the sphere looks like, particularly when it and the camera are on opposite sides of the inverse sphere center, and when the reflective sphere approaches a fractal zone. Stereo images would also be interesting to do.

When this research is finished, we will submit it for publication, along with some of the material presented here.

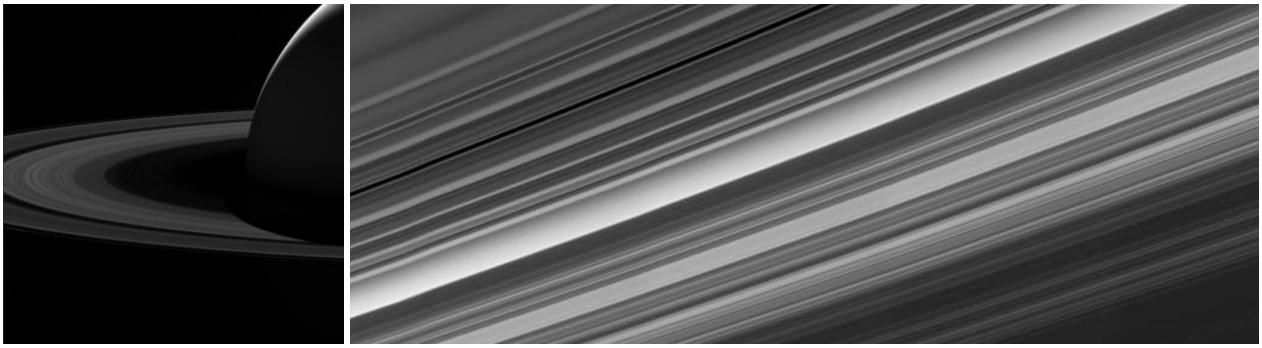
7.4 Vase Case Study

In Chapter 6 we modeled a vase, and rendered it with a number of materials and ray tracing techniques. Although we produced some nice images using our ray tracer and LuxRender, a lack of computing power limited what we could do.

8 Conclusion

We have presented in this thesis two different solutions, analytically for the inverse r sphere and numerically Runge-Kutta for the Luneberg spheres. Then we applied a different model i.e. glossy transmitter to produce frosted glass shader. We also developed a new Textured shader for both the color filter and the index of refraction, which produce nice images. We have tested our approach with another software and we can deal easily with complex objects such the vase we have modeled.

Finally, we have studied and implemented fractals formation when we have used inside camera and the inverse r sphere. Some other fractals famous examples: Mandelbrot set (Mandelbrot, 1982) and fractals showed in Saturn's rings. See below figures from NASA:



(a) (b)

Figure 58 *(a) Saturn's ring image from NASA. (b) Zoomed image of Saturn's ring.*

We can see how fractals are formed in Mandelbrot set in the figure below:

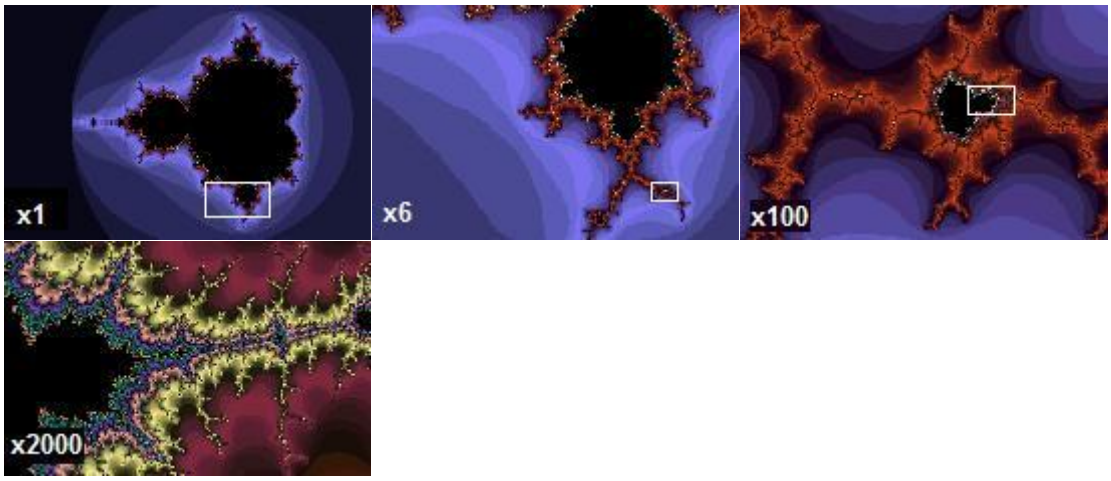


Figure 59: *Mandelbrot set zoomed x1 – x2000*

REFERENCES

Ament, M., Bergmann, C., and Weiskopf, D. (2014). Refractive Radiative Transfer Equation. *ACM Transactions on Graphics,* 33(2), Article #16.

Apodaca, A. A., and Gritz L. (2000). *Advanced Render Man: Creating CGI for Motion Pictures*. San Francisco: Morgan Kaufmann Publishers Inc.

Berger, M., Trout, T., and Levit, N. (1990). Ray Tracing Mirages. *Computer Graphics and Applications, IEEE,* 10(3), 36-41.

Blinn, J. F. (1977). Models of Light Reflection for Computer Synthesized Pictures. *Computer Graphics (Proceedings of SIGGRAPH '77)* , 11(3) 192-198.

Chandrasekhar, S. (1960). *Radiative Transfer.* New York: Dover Publications, INC.

Evans, J., and Rosenquist, M. (1985). *F = ma* Optics. *American Journal of Physics,* 54, 876-883.

Glassner, A. S. (1989). *An Introduction to Ray Tracing*. London, UK: Academic Press Ltd.

Handscomb, J. M., and Hammersley, D. C. (1964). *Monte Carlo Methods*. London: METHUEN and CO LTD.

Hecht, E. (2001). *Optics* (4th ed.). San Francisco: Addison-Wesley.

Jensen, H. W. (2001). *Realistic Image Synthesis Using Photon Mapping*. Wellesley, MA: A K Peters, Ltd.

Kajiya, J. T. (1986). The Rendering Equation. *Computer Graphics (Proceedings of SIGGRAPH '86),* 20(3), 143-150.

Lee, M. E., and Uselton, S. P. (1991). A Body Color Model: Absorption of Light Through Translucent Media. (J. Arvo, Ed.) *Graphics Gems II*, 277-282.

Luneberg, R. P. (1964). *Mathematical Theory of Optics.* Berkeley: University of California.

Mandelbrot, B. (1982). *The Fractal Geometry of Nature*. San Francisco: W. H. Freeman and Company.

Peachey, D. (1985). Solid Texturing of Complex Surfaces. *Computer Graphics (Proceedings of SIGGRAPH '85),* 19(3), 279-286.

Peachey, D. (2003). Building Procedural Textures. In *Texturing and Modeling: A Procedural Approach*, Third edition, edited by D. S. Ebert, F. K. Musgrave, D. Peachey, K. Perlin, and S. Worley, p. 7-94. San Francisco: Morgan Kaufmann Publishers Inc.

Perlin, K. (1985). An Image Synthesizer. *Computer Graphics (Proceedings of SIGGRAPH '85)*, 19(3), 287-296.

Pharr, M., Jakob, W., and Humphreys, G. (2017). *Physically Based Rendering: From Theory To Implementation* (3rd ed.). San Francisco: Morgan Kaufmann Publishers Inc.

Ralston, A., and Rabinowitz, P. (2001). *A First Course in Numerical Analysis* (2nd ed.). Mineola, New York: Dover Publications, Inc.

Rosser, D. (2012). https://github.com/danoli3/Multithreaded-Ray-Tracer

Shirley, P., and Marschner, S. (2009). *Fundamentals of Computer Graphics* (3rd ed.). Natick, MA, USA: A. K. Peters, Ltd.

Suffern, K. (2007). *Ray Tracing From The ground Up*. Wellesley, Massachusetts: A K Peters, Ltd.

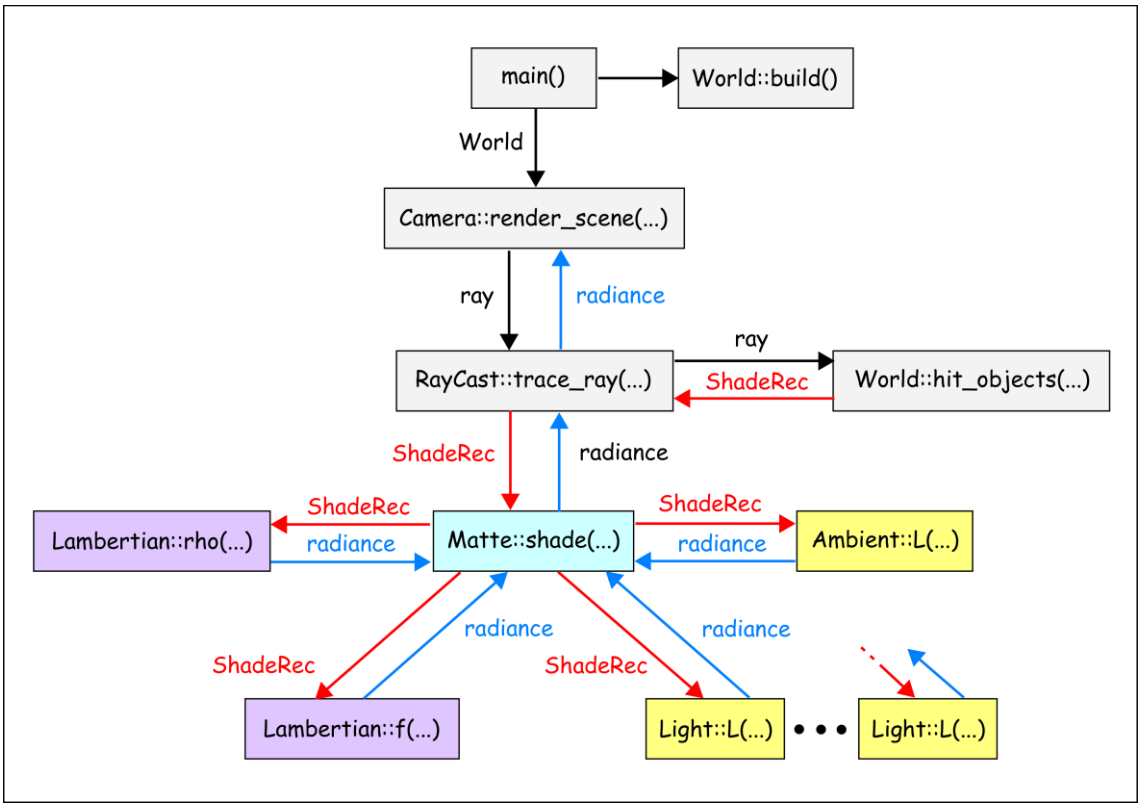
Suffern, K. G., and Getto, P. H. (1991). Ray Tracing Gradient Index Lenses. (N. M. Patrikalakis, Ed.) *Scientific Visualization of Physical Phenomena*, 317-331.

Whitted, T. (1980). An Improved Illumination Model for Shaded Display. *Computer Graphics (Proceedings of SIGGRAPH '85)*, 14(3), 110-116.

Zill, D. G., Wright, W. S., and Cullen, M. R. (2012). *Differential Equations with Boundary-Value Problems* (8th ed.). Boston, USA: Cengage Learning.

APPENDIX

A1 Data Flow diagram

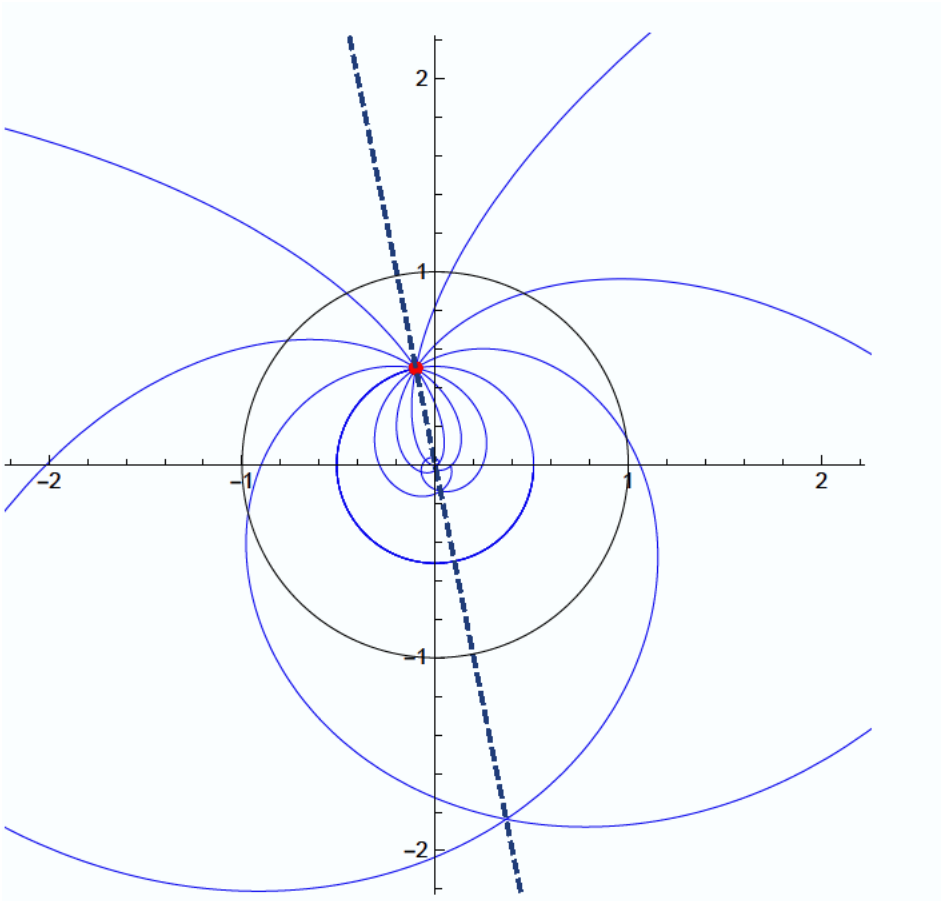


A2 Wolfram Mathematica

A2.1 Code

```
ClearAll["Global`*"]; (* clear all variables *)
Show
Table
e ={ex, ey}; (* eye 2D vector *)
d ={dx, dy}; (* direction 2D vector *)
R = 1; (* radius of the sphere *)
(* eye position *)
ex =-0.1;
ey = 0.5;
(* previous direction setup *)
(*dx =Cos[φ]; dy =Sin[φ];*)
(*dx=-2; dy=0.5b;*)(* e eye point component and d direction component *)
(*dx=1;dy=0;*)
(* psi s previous calculation, now psi is range from -Pi to Pi loop*)
(*ψs=ArcCos(ey dx - ex dy)Sqrtex^2+ey^2;*)
(* theta s *)
θs = ArcTan[ex, ey];
(* theta R *)
(*θR= θs+LogRSqrt[ex^2+ey^2]Tan[ψs];*)
(* avoid psi = 0 or Pi2 *)
If[ψs == 0, ψs = 0.00001, ψs];
If[Tan[ψs] === ComplexInfinity, ψs = 0.00001, ψs];
(* parametric equation in term of x,y and θ *)
x = Sqrt[ex^2 + ey^2] Cos[θ] Exp[Tan[ψs] (θ - θs)];
y = Sqrt[ex^2 + ey^2] Sin[θ] Exp[Tan[ψs] (θ - θs)];
(* polar equation in term of r and θ *)
r = Sqrt[ex^2 + ey^2] Exp[Tan[ψs] (θ - θs)];
(* mathematica ParametricPlot *)
plotXY = ParametricPlot[{x, y}, {θ, θs, Sign[ψs] 2 Pi},
PlotStyle → Directive[AbsoluteThickness[0.01], Blue]];
(* mirror cut line *)
(*mirror=ParametricPlot[{(ey/ex) t,(ey/ex )t},{t,-5,5}];*)
mirror = Plot[(ey / ex) t, {t, -5, 5}, PlotStyle -> Dashed];
(* mathematica PolarPlot *)
(*plotR = PolarPlot[r,{θ,θs, θR},
PlotStyle→Directive[AbsoluteThickness[0.01],Blue]];*)
(* draw start point ps when theta = thetas *)
Block[{θ = θs}, ps = Graphics[{Red, PointSize[0.015], Point[{x, y}]}]];
Show[plotXY, ps, mirror],
(* loop for psi value, used offset = 0.01 *)
ψs, -Pi, Pi, Pi 8,
(* mathematica circle draw *)
Graphics[{Circle[{0, 0}, R]}] ,
AspectRatio → 1, PlotRange → {{-2, 2}, {-2, 2}}
```

A2.2 Plot



A3 C++ Source Code

A3.1 RungeKutta.cpp

```
double h = 0.01;
//            double h = 0.02;
double sx, su, sy, sv, sz, sw;
double k11, k12, k13, k14, k15, k16;
double k21, k22, k23, k24, k25, k26;
double k31, k32, k33, k34, k35, k36;
double k41, k42, k43, k44, k45, k46;

sx = ray.o.x;
sy = ray.o.y;
sz = ray.o.z;

su = f(sx, sy, sz) * ray.d.x;
sv = f(sx, sy, sz) * ray.d.y;
sw = f(sx, sy, sz) * ray.d.z;

/*su = eta * ray.d.x;
sv = eta * ray.d.y;
sw = eta * ray.d.z;*/
```

```
        bool condition = true;
        vector<Point3D> ray_points;              // local array of points
        Point3D position(sx, sy, sz);            // initial position on the
                                                 transmitted ray
        ray_points.push_back(position);    // store hit point in the array
```

A3.2 Inverse r sphere

```
RGBColor
Whitted::trace_ray(const Ray ray, Light* lt, const int depth) const {

    if (depth > world_ptr->vp.max_depth)
        return (black);
    else
    {

        double ex, ey, ez, dx, dy, dz;
        double cx = 0;
        double cy = 0;
        double cz = 0;

        Point3D e(ray.o);                  //eye
        Vector3D d(ray.d);                 //direction
        Point3D c(cx, cy, cz);

        Vector3D du = d / d.length();              //unit vector d

        double s = (e - c).length();

        Vector3D b = (e - c) / s;                  //vector b

        Vector3D w = (b^du) / (b^du).length();           //vector w

        Vector3D bp = w^b;                                      //vector bp

        double psi = atan(sqrt(1 - pow(bp*du, 2)) / (bp*du));  //psi

        double thetas = asin(((e - c)*bp) / (e - c).length()*bp.length());
    //theta S

        double thetar = thetas + log(R / s) / tan(psi);
        //theta R

        //elhoni
        double theta = -2*PI;
        double r = 0;
        double l;
        Vector3D p;

        while (theta < 2*PI)
        {

            // general spiral equation
            r = s* exp(tan(psi)*(theta - thetas));

            // if r gratear than radius break
            if (r >= R) { r = R; break; };
```

```
                    // below steps for computing l componenet and its
dervivative

                    l = -s * cos(thetar) + sqrt(s*s*cos(thetar)*cos(thetar) +
r*r - s*s);

                    double lc = l*cos(thetar);

                    double ls = l * sin(thetar);

                    p = lc*b + ls*bp + e;              //point equation 3D

                    // inside sphere point center location
                    Point3D cis(2.0, 0.0, 0.0);

                    // if point square - inside sphere center square less than
radius return red
                    if (((p.x*p.x) - (cis.x*cis.x)) <= 1)
                           return RGBColor(0.7, 0.3, 0.2);

                    theta += 0.1;

             }

             l = -s * cos(thetar) + sqrt(s*s*cos(thetar)*cos(thetar) + r*r -
s*s);

             double lc = l*cos(thetar);

             double ls = l * sin(thetar);

             p = lc*b + ls*bp + e;              //point equation 3D

             // dervative for l component used for computing direction out rays
             double Dl = s*sin(thetar) - (s*s*cos(thetar)*sin(thetar)) /
sqrt(r*r - s*s + s*s*cos(thetar)*cos(thetar));

             double Dlc = -l*sin(thetar) + Dl*cos(thetar);

             double Dls = l*cos(thetar) + Dl*sin(thetar);

             Vector3D pd = Dlc*b + Dls*bp;             // out rays vector
direction

             Ray test_ray;               // out put ray
             test_ray.o = p;
             test_ray.d = pd;

             //Point3D hit_point(sr.w.objects[0]->hit(test_ray));
             ShadeRec sr(world_ptr->hit_objects(test_ray));

             // set up sr
             if (sr.hit_an_object) {
                    sr.hit_an_object = true;    // we may not need this
                    sr.depth = depth;
                    //sr.ray.o = test_ray.o;          //computed hit point of
the ray on the inside of the sphere
                    //sr.ray.d = test_ray.d;          //computed ray direction
at the hit point
                    //sr.ray = test_ray;
                    //sr.normal = -sr.normal;
                    //sr.hit_point = test_ray.o;
```

```
                    //sr.normal = world_ptr->objects[0]->get_normal(test_ray.o);

                    //sr.material_ptr = world_ptr->objects[0]-
>get_material();//the dielectric material

                    return (sr.material_ptr->shade(sr));     // the standard
recursive call
               }


               else {


                    return (world_ptr->background_color);
               }


       }


       }
```

A4 Autodesk Maya

تتبع الشعاع في المواد الشفافة خلال تغير خواصها الطبيعية

إعداد

أحمد علي الصادق الهوني

المشرف

د. محمد العماري

المشرف المساعد

د. كيفين سفرين

الملخص

يتم محاكاة المواد الشفافة في تتبع الأشعة باستخدام أشعة الضوء المنعكسة والمرسلة. في هذه الرسالة نقدم تقنية بسيطة لتتبع المواد الشفافة للأشعة حيث يختلف لون المرشح داخل المواد حسب الموضع. يمكن أن يكون انعكاس السطح ونقله براقًا أو لامعًا. يعتمد ذلك على تتبع الأشعة ويتيد ستايل وتتبع المسار. نقوم أيضًا بتتبع المواد الزجاجية حيث يختلف مؤشر الانكسار مع الموضع. ينتج عن ذلك مسارات ضوئية منحنية يجب حسابها عدديًا في بعض الحالات. نستخدم تقنية تتبع الشعاع مع عدسات لونبيرق متناسبًا مع r / 1، و r هي المسافة الشعاعية من المركز. نحن نسمي هذه المجالات معكوس. نقوم بتحليل مسارات الأشعة الضوئية داخل هذه المجالات ، والتي هي لوالب لوغاريتمية ، وتتبعها الأشعة من الداخل باستخدام كاميرا عين السمكة. تتيح هذه الكاميرا تصوير الأشعة في جميع الاتجاهات من موقع الكاميرا. لقد وجدنا أن الصور الناتجة تحتوي على فراكتل واحد أو أكثر ، وقد سمح لنا تحليلنا بفهم كيف تشكلت الفركتلات. قدمنا عددًا من الصور لتأكيد وجود صور النمطي هندسي متكرر.

كدراسة حالة ، قمنا بتصميم مزهرية في المايا مع شبكة مثلث عالية الدقة ، ونجعلها في جهاز تتبع الأشعة لدينا مع مجموعة متنوعة من المواد وتقنيات تتبع الأشعة. وتشمل هذه المواد الارسال العازلة والمصقول ، ولون مرشح لون قوس قزح ، وتتبع المسار لإنتاج المواد الكاوية الملونة.

هناك المزيد من العمل للقيام به مع المجالات معكوس. نريد أن نتعقبهم عندما يكون مؤشر الانكسار على السطح أكبر من واحد ، وله قيم كبيرة بشكل تعسفي. نريد أيضًا وضع كرة عاكسة داخل كرة عكسية ، ومعرفة الشكل الذي يبدو عليه عندما نتبعه باستخدام الكاميرا من الداخل والخارج.

تتبع الشعاع في المواد الشفافة خلال تغير خواصها الطبيعية

قدمت من قبل :

أحمد علي الصادق الهوني

المشرف:

د. محمد العماري

المشرف المساعد:

د. كيفين سفرين

قدمت هذه الرسالة استكمالا لمتطلبات الحصول على درجة الماجستير في علوم الحاسوب.

جامعة بنغازي

كلية تقنية المعلومات

يونيو 2019