Colors in Flux: Rendering Iridescence Arising from Single-Layer Thin Films

A Thesis
Presented to
The Division of Mathematical and Natural Sciences
Reed College

In Partial Fulfillment
of the Requirements for the Degree
Bachelor of Arts

Mercy M. Bhakta

May 2020

Approved for the Division
(Computer Science)

James D. Fix

Acknowledgements

Writing a thesis is a good amount of work, and requires a great amount of support and patience from others. I'd like to take the time to thank my parents and grandparents for their endless encouragement and support. While my dad never had the chance to see my finished thesis, he would have been proud. A thank you goes to my mom for her strength and love in these turbulent times. My sister gets a special shout-out for actually reading the whole thing, thanks Raveena! A big thank you to my family, friends, the CS Gems, and everyone who told me I could make it through Senior year. Thanks to Lilian for all the "thesis (work) parties" and for your friendship! To my professors at Reed, a huge thank you for the enthusiasm, support, and knowledge you have given me. I would be remiss to ignore my bosses at CUS, whose kindness and encouragement were a big part of my time at Reed. Also, a thanks must go to Britt Hoover, whose kindness helped me through this semester. And finally, if you are reading this, thank you for taking the time to do so.

Table of Contents

List of Figures

Abstract

The problem of representing interference phenomenon such as iridescence is challenging due to the photon-based models used in computer graphics. This, and other effects due to wave optics are phenomena not well-represented in computer graphics. My thesis surveys and synthesizes mathematical models to render iridescent phenomenon due to the wave nature of light and its interactions in single-layer thin films on surfaces. This derivation integrates with ray tracing techniques, and a possible implementation in pseudocode is also presented.

Dedication

To my parents-

mein tumaru naam roshan kari rahi chhu, dheeme dheeme.

Introduction

I never paint dreams or nightmares. I paint my own reality. - Frida Kahlo

Kahlo's quote frames humanity's fascination with representation in her own light: beset by pain after a bus accident, her surrealist paintings are hardly fantasy, but rather reflect her own personal truths. In this way, through representation, humanity discovers a sense of grounding and fascination. This representation is not necessarily just of self, but of the world around us. The stories and images we create allow us to explore and better understand the world around us. And, of course, to entertain.

> "[Evolution of humanity] is also a story of how we became a species obsessed with creating artistic experiences with which to amuse, shock, titillate, and enrapture ourselves, [...] from firelit caves to the continuous worldwide glow of television screens." [5]

To render a person, animal, or thing is to pay homage to their existence from one's unique perspective, or perhaps, as a visual mnemonic or guide for the future. We can only speculate the reasons why ancient peoples dipped fingertips in pigment and smeared stories onto the rocky walls. These two-dimensional (2D) representations first seen in cave paintings naturally evolved into a preoccupation with reproducing three-dimensional (3D) objects accurately onto 2D surfaces. Nowadays, 3D representation is commonplace and can be seen in movies, furniture catalogues, video games, and more.

While there are many ways to reproduce visuals from the 3D world around us onto a 2D medium such as a screen, one (perhaps unconventional) method of interest to this thesis is using a pinhole camera. A pinhole camera produces an image from light passing through a small hole, typically the size of a pinprick. Historically, pinholes were a natural *camera obscura* to view eclipses and study light:

> "Many myths have been retold of ancient peoples seeing pinhole images inside tents, darkened rooms, and the like, so it would seem that a living knowledge of pinhole images has occurred for tens of thousands of years."

[18]

This natural phenomenon led to the construction of the modern pinhole camera.

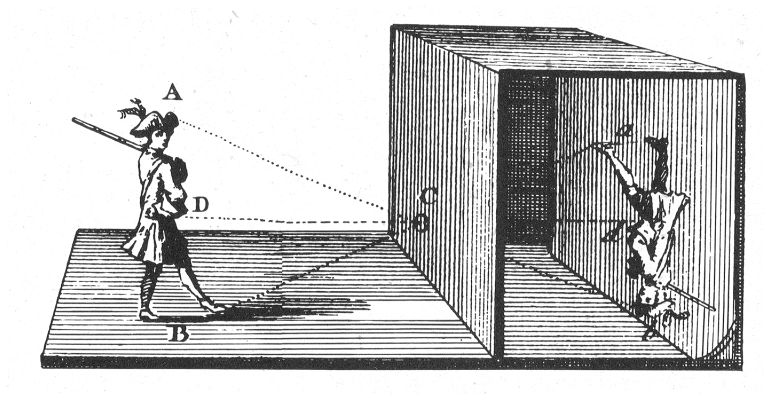


Figure 1: A pinhole camera without film. (Camera Obscura)
Image taken from Wikimedia Commons.

The camera worked by letting rays of light enter a pinhole on one side of an enclosed box. The side opposite the pinhole had film which recorded incoming light rays. Because of the nature of light entering the pinhole at certain angles and striking the film, images produced were upside-down from reality, but otherwise the process was a quick way to capture moments in time.

Fundamentally, images produced from pinhole cameras rely on light and shadow to differentiate features. Through an artist's perspective, 3D forms are "created" by the interplay of shadows and light; as the artist Nicolaides put it, "The folds in a distant range of mountains may be distinguished most clearly when the shadows appear" [13]. While we need light in order to detect form, we also need shadows for the brain to perceive depth cues. The pinhole camera, with its seemingly simple setup, does a worthwhile job of rendering the world around us. The premise of a pinhole camera, letting rays of light create representation of the world, has been co-opted in computer graphics in a technique called ray tracing.

Ray tracing is a modern way of depicting the world around us. Well, not exactly the *physical* world around us, as computer graphics seeks not only to accurately render representations, but to also create an accurate model of some world. The technique of ray tracing creates images by modeling light rays in a virtual scene and recording those rays, using the same structures as a pinhole camera does. Light in the virtual scene, as in the real world, has many complexities when encountering a surface. When we look out into the world, we often take for granted light's interactions. We usually don't think of the intricacies of light reflecting and absorbing, or wavelengths shifting. Instead we live life, taking in the rich hues of a sunset on the horizon, or the shimmer and beauty of a hummingbird's feathers.

Today, in computer graphics, such effects as light reflection and transmission are well-modeled. Modeling light phenomena based on wave optics, on the other hand, is a more difficult issue. Iridescence[1] is one such phenomenon. Formally, iridescence is the perceived changing of color based on light and the position of the viewer. Different angles can give view to different vivid colors. Think of the rainbows shifting and swirling inside of soap bubbles floating on the great lawn, or the jewel-toned armor of a scarab beetle. Beauty is indeed in the eye of the beholder when it comes to iridescence.

A viewer of such effects might begin to wonder what causes iridescence and why only certain materials exhibit the phenomenon. There are two main causes of iridescence, thin-film interference and diffraction due to microstructures. These are described in more detail later on. For now, know that these are due to light's interac-

[1]Also known as goniochromism.

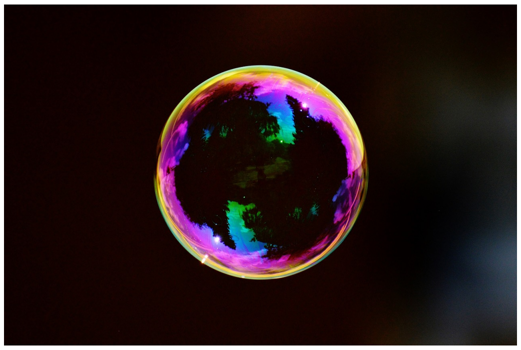


Figure 2: A soap bubble displaying iridescence.
Photo taken by Alexas_Fotos on Pixabay.com

tions with the surface rather than pigmentation. When we view the striking blues of a Morphos butterfly, we may never realize that in fact the pigment of their wings is brown. It is the light that changes perceptions and enraptures the viewer. However, rendering these "interference colors" (as opposed to color arising from pigment) is challenging within the current framework of computer graphics. While modern-day computer graphics has gotten better at depicting light phenomena, it still has room to improve.

This thesis seeks to provide a shading model to implement iridescent shading for ray tracing. In order to do so, Chapter 1 focuses on the necessary basics for image rendering in computer graphics, with concepts ranging from mathematical structures for scene geometry to the physics behind light and color. Chapter 2 builds upon these basics by introducing the notion of ray tracing and illumination models for physically-plausible image creation. Chapter 3 delves into iridescence: we take a look at the science behind it, examine the causes and setup necessary for it to occur, and then provide our own formulation for the phenomenon.

Chapter 1

"Painting" with Light

Viewing representations of the world around us on a screen can enrapture and inspire viewers, but how does one move from the role of consumer of such images to creator? This chapter delves into the many abstractions used in computer graphics for simulating and rendering the physical world. We begin with a review of mathematical concepts commonly used in the field, using those to construct geometries within the world and to model light. We then show how light can interact with surfaces in our virtual world, covering reflection, transmission, and diffraction on diffuse and specular surfaces. This, in turn, sets the stage for calculating color based on the energy of light and the surface material of the object. To calculate color, we compare different shading models that approximate the appearance of light's effects before examining *physically-based* shading models that accurately model the energy transfer of light. This will later allow us to accurately render images. With this framework in mind, let us begin.

1.1 A Bit O' Math and Creating a Virtual World

We can define our virtual world with a three-dimensional Cartesian coordinate system modeling Euclidean geometry. The 3D coordinate system in $\mathbb{R}^3$ has the familiar x and

y axes from the 2D Cartesian system, as well as a z axis for depth [8]. For its usage in graphics research, the orientation of the coordinate system usually follows a right-hand rule, with the positive z-axis pointing forward, toward the viewer. Intuitively, one can raise their right hand with the palm facing themselves. Keeping their index finger up and thumb out represents the positive y and x axes, respectively. Pointing their middle finger towards themselves, they can see which way the positive z-axis (represented by their middle finger) is oriented.

Now that we have an empty world full of possibilities, we can create points to reference locations in relation to one another. These are stored as (x, y, z) coordinates, with $x, y, z \in \mathbb{R}$. Offsets between points can be depicted by vectors. These have a magnitude (length) and a direction. We can represent vectors in a 3-dimensional space with a 3×1 or 1×3 matrix $[x, y, z]$, where x, y, and z are the components of the vector with respect to the coordinate system [10]. The length of a vector, v, is represented as $|v|$. This is also known as the *magnitude* of a vector and can be calculated using the Pythagorean Theorem: $|v| = \sqrt{x^2 + y^2 + z^2}$.

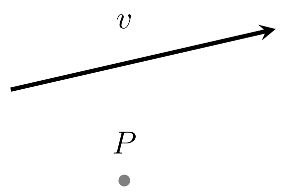


Figure 1.1: A vector, v, and point, P.

Vectors, as rigorously defined in mathematics, have defined operations and properties based off the vector space they are in. They can be multiplied by a scalar value, added and subtracted together, and negated. Scalars are single elements of a field, F, as opposed to vectors which have multiple components to represent a direction. Scalar multiplication takes a scalar value and multiplies it to each component in a vector. Vector addition is a special form of addition just for vectors, in which corresponding components are added to each other. Visually, if we take two vectors, u and v, and add them together, we can take the vector v and place it at the head of vector u. The vector that starts at u and ends at v is the result of the addition. Each vector, v in the vector space has an additive inverse $-v$, which visually is just the vector flipped in the opposite direction. This is especially useful for visually representing vector subtraction: we can negate the second vector and add it to the first.

One may note that there is no such vector multiplication akin to vector addition. Instead, the cross product and dot product are defined and used. The cross product of two vectors, $u \times v$, is the vector that is at a right angle to both vectors and has a magnitude equal to the area of the parallelogram formed by both vectors. The dot product of two vectors multiplies the length of each vector by the $\cos\theta$ between the two. This is written as follows, $u \cdot v = |u||v|\cos\theta$. If v is a unit vector, this corresponds to the length of the projection of u onto v. The projection of a vector u onto v is the vector v scaled by the length of the component of u that is parallel to v. This is the length of the side adjacent to θ when forming a right triangle from u to v using a perpendicular line to v. Vectors themselves can also be *normalized*, where each component is divided by the magnitude of the vector. This changes the vector to be of length 1, and is commonly used in computer graphics for calculations.

Together, points and vectors give us a wider range of operations. For example, we can define point subtraction between two points, $P - Q$, as the vector starting at Q and ending at P. Point-vector addition, $P + v$, starts at a point, P and is the point Q given by moving in vector v length and direction. Geometrically and intuitively, this makes sense.

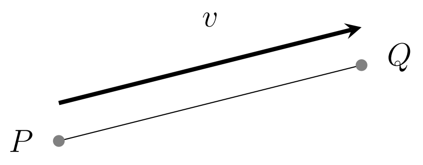


Figure 1.2: $Q - P$ gives the vector v, and $P + v$ gives the point Q

Having points and vectors in our virtual world now gives us an adequate way of representing light. We think of light as a ray, which is quite a bit different from the colloquial usage of the term "a ray of light." In this usage, we imagine the energy of light as a bundle, or photon, that is emitted from a light source and can travel in a straight path. This can be modeled using the mathematical notion of a ray.

Definition 1. A **Ray** has a set origin point and continues infinitely in one direction. We can mathematically define a ray as the set of all points $R = \{P + tv \mid t \in [0, \infty)\}$ for origin point P and direction vector v.

Now that our virtual world can potentially contain rays of light, we need something for that light to shine on. In order to model real-life phenomena, and interactions between light and matter, we must first model objects in a three-dimensional world. Below, we explore the representation of simple objects such as planes and spheres.

Modeling Planes

A plane is a 2-dimensional surface that extends infinitely. We can model this with a point on the plane, P, and a normalized vector pointing away from the surface, n. This is known as the surface normal. Recall that the dot product of two vectors that are perpendicular to each other is 0. Using this, we can define a plane by its points. The whole plane is as follows.

$$\{Q \mid (Q - P) \cdot n = 0\}$$

where (Q-P) is the vector on the plane from P to Q, and n is the surface normal perpendicular to that vector [8].

Modeling Spheres

We can model a sphere by starting simpler. Recall the equation for a circle in $\mathbb{R}$: $x^2 + y^2 = r^2$, where x and y are the coordinates of the circle center, and r is the radius of the circle. In our 3D world, we can represent a sphere as $x^2+y^2+z^2=r^2$, centered at the origin. In terms of points, we can represent this as $\{P \mid |P-O|^2=r^2\}$, where O is the center of the sphere, and P is a point on the surface of the sphere.

1.1.1 Object-Ray Intersections

So far, we have ways to model light and objects, but how do we know if an arbitrary ray of light reaches those objects? According to the science of the human visual system and physics, when light photons are emitted from a source, they travel in a straight path, bouncing around and striking objects (this is a simplified view, complexities of light are explored later on). Photons that eventually reach our eyes are how we see. In our world we can calculate the intersection between simple objects and a light ray in order to determine whether light reaches the object.

Plane-Ray Intersections

A ray and a plane can have two possible intersection types, the first being a direct intersection through the plane; the second with the ray parallel to the plane, i.e. contained within the plane. It can also, of course, potentially not intersect at all. To determine if the light ray actually intersects with the plane, recall the parameterized equation for a ray starting at point P with direction vector v: $P+tv$, and recall the property of a point X on the plane defined by point Q and surface normal n: $(X-Q)\cdot n=0$. For the light ray to intersect, the following must hold:

$$((P+tv)-Q))\cdot n=0$$

This equation expresses that a light ray will intersect a plane if it is orthogonal to the surface normal. If the ray is within the plane, there are an infinite number of solutions.

Sphere-Ray Intersections

A ray of light can intersect with a sphere in two ways as well: it can intersect at one point (when the ray is tangential to the sphere or when the light source is inside the sphere), or at two points (when the ray is passing through the sphere). We can calculate the intersection either from a geometric viewpoint or algebraically. Geometrically, we can check the minimal distance between the center of the sphere and the ray. If the distance is less than the radius, then we have an intersection. Algebraically, we can use the equation for a sphere centered at point O, $|X - O|^2 = r^2$, then use the fact that any point can be represented through the parameterized equation of a ray, $X = P + tv$. Because of this, we can plug in the equation for X as follows, $|(P+tv)-O|^2 = r^2$ and solve for t [20]. This ends up being the solution to a quadratic equation, so there is either 1 solution, 2 solutions, or none, which corresponds nicely to the possible intersections.

Now that we know when light can reach an object, what happens when it does? We turn to physics for some answers about light's interactions with the surface of objects.

1.2 The Nature of Light

As previously mentioned, artists throughout time show that human beings perceive form through light and shadow. Without the two, there would be no depth to life. Everything would instead appear flat, akin to a line drawing. Realms away, researchers in physics such as Max Planck and Albert Einstein tell us that light is made up of photons, with a photon defined as a "bundle of electromagnetic radiation"[12]. We can assume that these photons, or "light particles" act as particles do. In photorealistic rendering, a subfield of graphics that falls within the intersection between art and physics, it is light's interaction with matter, as well as modeling the substance of that matter, that is of interest to computer graphics researchers. For basic rendering, it is sufficient to know that light photons can both be reflected upon striking the surface of an object, or they can be transmitted through that object. Both of these interactions rely on the assumption that light acts as a particle. Below, we delve a little deeper into light's interactions with matter, as well as how those interactions are modeled in computer graphics.

1.2.1 Reflection

When a photon of light hits a certain surface it can "bounce" off, much like a ball bouncing off a wall. This is known as a reflection. Marion & Hornyak describe the process as "redirected (and perhaps altered) light that originated elsewhere" [12]. The exact mechanics of how light reflects depends on the type of surface it strikes.

Specular Surfaces

We begin our journey into reflections with the simplest type: light bouncing off of a specular (mirror-like) surface. Perfectly specular surfaces, such as a mirror[1], reflect light with outgoing rays of light (light that bounces off a surface) at the same angle to the surface normal as the angle of the ray of incident light. The surface normal at a point P is the direction the surface is oriented towards at that point. In the figure below, the angle of *incidence* is θ_1 and the angle of *reflectance* is θ_2. For specular surfaces, $\theta_1 = \theta_2$ [12]. Visually, seeing a direct reflection results in bright highlights on the surface that are the same color as the light source.

[1]Of course in real life mirrors are **not** perfect specular surfaces, due to micro-imperfections.

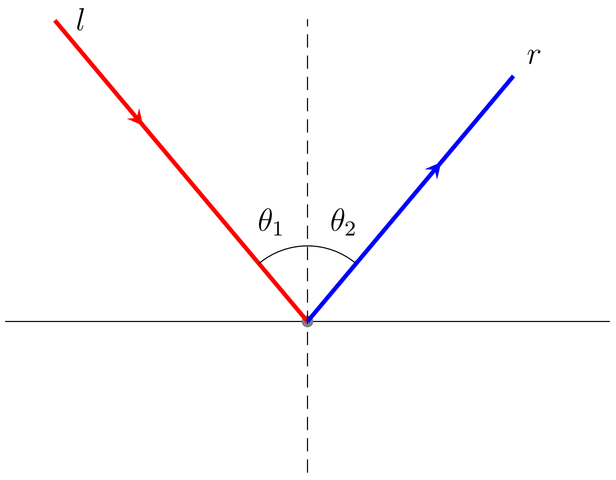


Figure 1.3: Reflection off a Specular Surface

Diffuse Surfaces

When a perfectly diffuse surface[2] encounters a beam of light with many photons, something unexpected happens. Instead of bouncing off in a certain direction as happens for a reflective surface, photons striking a diffuse surface will scatter seemingly in all directions, with equal energy, causing some light to reach the viewer [7]. When describing this reflection in all directions, we can think of this as an energy distribution over a hemisphere with the surface point the photons strike as its center [7]. This distribution of light results in a matte-like appearance of the surface, with color remaining approximately the same no matter where the viewer stands.

The amount of light that reaches the surface, and can therefore be reflected, is proportional to the $\cos\theta$ of the incident light ray and the surface normal [7]. A light ray directly in-line with the surface normal will have a $\cos\theta$ of 1, and a light ray that is parallel to the surface (i.e. perpendicular to the normal) will have a $\cos\theta$ of 0. This fits our intuition of how received light changes the brightness of a surface: a light reaching a surface head-on will appear brighter than one at an angle. When light is close to parallel to a surface, but not quite parallel, light spreads out along

[2]Such surfaces are known as *Lambertian* surfaces. As with perfectly specular surfaces, these also do not occur naturally.

the surface. A light beam parallel to a surface doesn't directly reach the surface at all. In that case, the surface does not receive illumination.

Once light does reach the surface, scattering in "all" directions occurs due to the roughness of a surface. Even though diffuse surfaces such as paper do not seem rough, they are at the microscopic level. These variations in the height and angle of the surface can be represented with microfacets. For diffuse reflections, photons scatter off of the microfacets which are oriented at different angles.

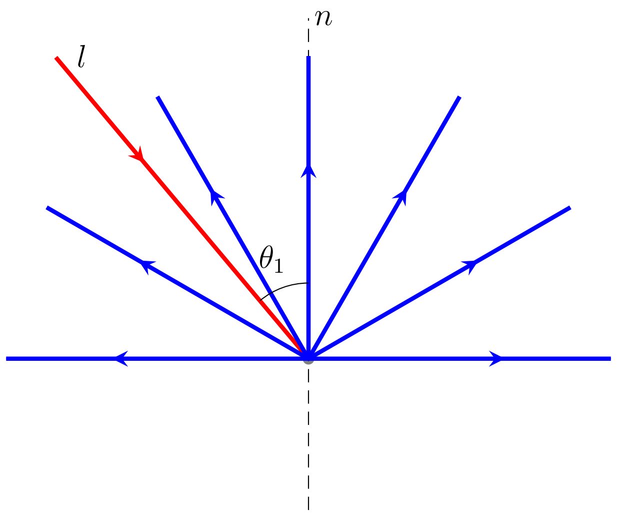


Figure 1.4: Reflection off a Diffuse Surface

While perfectly diffuse surfaces may not exist in nature, they are a useful approximation for how light interacts with many surfaces. It is especially useful for synthesized objects, such as walls, paper, fabric, and other human-made materials.

1.2.2 Transmission

Light photons can also pass through the surface of an object. This is known as transmission of light. Some mediums, such as oil or water, cause light to slow down as compared to the speed of light in air. When light is transmitted at an angle to a surface, with a change in speed, it undergoes refraction, or "bending" of light at an angle as seen in Figure 1.5. The solid grey line represents a change in interface, and

θ_2 is the *angle of refraction* [12]. Examples of this include a straw in a glass of water;

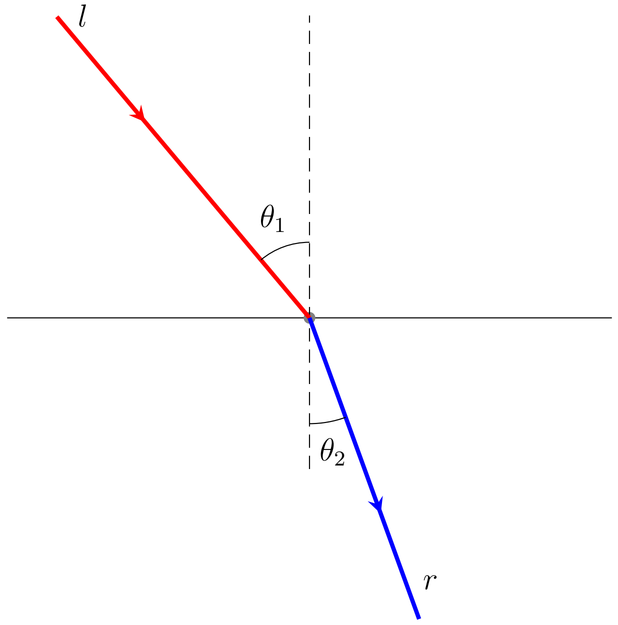


Figure 1.5: Refraction of light

the straw appears to bend as it passes through the water. This is due to the fact that light travels through different mediums at different speeds. In air, the speed of light is defined as the constant $c = 2.99 * 10^8$. In oil, light travels much more slowly, and the bending is explained in the transition from the faster speed of light in air to the slower speed of light in oil. The degree to which it bends can be calculated with the index of refraction, typically denoted as η in physics.

Definition 2. The **Index of Refraction** (η) of a medium is c/v
where c is the speed of light in meters per second, and v is the velocity of light in the medium.

Light changes directions in accordance to the index of refraction in the new medium whenever it encounters an interface with a slower speed. This can be expressed through the Law of Refraction.

Definition 3. The **Law of Refraction**, or **Snell's Law** states that

$$\eta_1 \sin\theta_1 = \eta_2 \sin\theta_2$$

for η_1 and θ_1 in the first medium, and η_2, θ_2 in the second [12].

1.2.3 Diffraction

However, the story of light being represented as a photon does not quite capture the full picture. Light also behaves as a wave. This is known as the wave-particle duality of light. One property of light as a wave is the ability to diffract. Diffraction is the

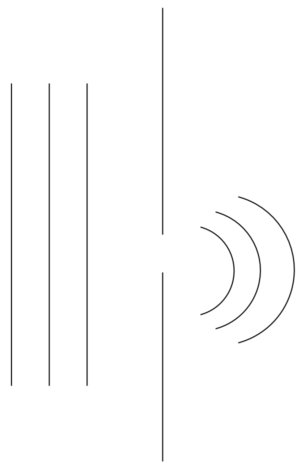


Figure 1.6: Diffraction of light

breaking and bending of a wavefront (the representation of a peak on a wave, often shown as a straight or curved line) into waves, each with their own wavefronts, with infinite potentials to break into new waves. This occurs when "waves interact with obstacles or apertures with finite size" [12]. Visually, this phenomenon is present in water waves rippling between logs on a lake, perhaps on a cold, clear day.

One may wonder, how do we relate all of these interactions to 3D images on laptop screens? The short answer is that rendering techniques such as ray tracing, along with shading models, are used to determine colors for the array of pixels that comprise a screen.

1.3 An Overview on Color

Using ray tracing to calculate the color of a surface begs an obvious question: how do we represent colors in our virtual world? As usual, we take inspiration from the world around us. Biologically, humans perceive colors when incoming light strikes certain photoreceptor cells called cones in the retina of the eye. Physics and chemistry tell us that light is made up of a spectrum of wavelengths. Humans can only see a certain range of the electromagnetic spectrum. This range is known as the visible spectrum, and it ranges from a wavelength of around 400 to 800 nanometers, or around $7 \cdot 10^{14}$ to $4 \cdot 10^{14}$ Hz for frequency. Wavelength (λ) is usually used to talk about color, since $\lambda = \frac{c}{v}$, where c is the speed of light in a vacuum, and v is frequency in hertz (Hz). Certain materials can absorb or reflect certain frequencies of light; this determines the color that we see. Depending on the material, only photons of a certain frequency can be absorbed. The rest are transmitted through the material or reflected. For example, red apples reflect photons of frequencies that roughly correspond to red in the visible spectrum.

Despite humans having a nearly set range of visible color perception, colors are, in fact, a subjective experience. Differences in environment and lighting all affect the perception of a color's brightness and chromaticity. Because of this, attempts at standardization for color have been made, based off of more concrete factors. Recall that the retina in our eyes contain photoreceptor cells, of which there are two types: rods and cones. Rods in our eye detect brightness of light, while cones detect color. There are three types of cones (long, medium, and short) which each form a distribution for detecting light, and these roughly match human perception of color ranges over red, blue, and green. Visually, colors "can be reproduced by combining various strengths of light in red, blue, and green wavelength bands" [11]. Using this fact, the CIE (Commission Internationale de L'eclairage, or International Commission on Illumination) created a model for standardizing an average perception of these primaries in 1931 with their Standard Observer model[3]. They did so by having viewers visually match colors using varying degrees of monochromatic red, green, and blue light [21]. Participants changed the intensity of the red, blue, and green light in order to match the target wavelength's color. This was then repeated for multiple wavelengths to form a distribution. They found that different spectral distributions can give rise to the same color, and that some green-blue colors were not reproducible, unless red was "subtracted"[4] [7]. The results of this study can be seen in Figure 1.7. The graph shows the intensities required of each monochromatic red, blue, and green light to reproduce the color perceived at each wavelength interval. Because of the "subtraction" causing negative intensity values, the CIE decided to linear transform the RGB values to $\bar{x}, \bar{y},$ and $\bar{z}$ primaries, which are non-negative. These are within a XYZ color space, rather than a RGB color space. This model was incredibly useful, allowing researchers to move from the subjective space of color perception into that of numerical cohesion...I mean objectivity.

These numerical values that are obtained from spectral distributions in the model are known as tristimulus values, represented by X, Y, Z. Respective $\bar{x}(\lambda),\ \bar{y}(\lambda),\ \bar{z}(\lambda)$ are used to calculate these.

After being obtained from a spectral distribution, the XYZ tristimulus values are then converted to Red, Green, and Blue (RGB) values. These are stored in a color vector: $[R, G, B]$. The values range from [0.0 - 1.0] for most graphics systems, and then are converted to the range [0-255] in order to be displayed on a screen.

In order to calculate the color and brightness of an object, we need to know the

[3]The performed study limited participants to a 2° field of view. A later study done in 1964 used a field of view of 10° and is currently recommended for use instead.

[4]Subtracted in this case meaning added to the target color.

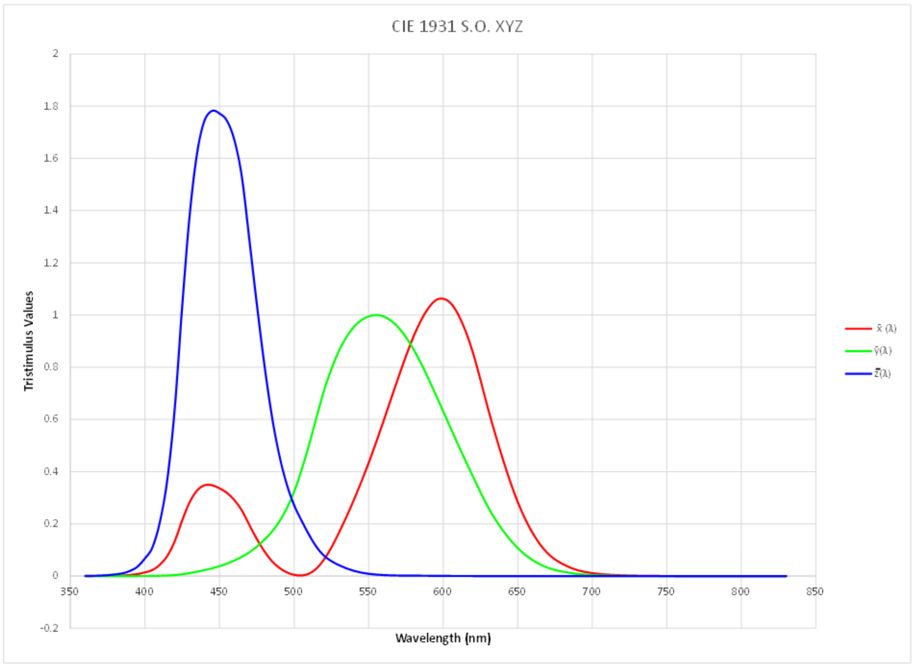


Figure 1.7: The CIE 1931 Standard Observer

amount of energy transferred to it. To do so, various shading models are used.

1.4 Basic Shading Models

Shading, as Shirley and Marshner put it, "paints the surface with light" [20]. Although less poetic, in *Principles of Digital Image Synthesis*, Glassner is a bit more exact in the definition he brings to the table: shading is "computing light that leaves a point either by self-emission or propagation" [7]. Essentially, a point on a surface can either emit light or propagate it in some way, either through transmission or reflection. If a surface is visible, light is reflecting back to the eye. How much light is reflected depends on the type of the surface as well as other factors, such as the angle of incoming light. Recall light's interactions with diffuse and specular surfaces. In real life, most surfaces exhibit a mix of the two. Earlier shading models did not attempt to model the complexities that arise for light or the energy of light, and are instead approximations of how light interacts with surfaces in the real world. These models tend to be made of three components: one for ambient lighting, one for light reflecting from diffuse surfaces, and one for light reflecting from specular surfaces. In addition to these, there are also non-photorealistic shading models, meant to provide an aesthetic perspective to graphics, rather than a physically accurate one, but we won't be discussing them at length in this paper.

1.4.1 Ambient Lighting

When viewing a poorly-lit room, there is rarely anything that is completely unilluminated, even if the light source is unable to directly reach all areas of the room. This is because light reflections can bounce several times before attenuation of brightness renders the light undetectable to the human eye. These indirect reflections cause the room to be *ambiently-lit* or have ambient lighting.

Graphics researchers approximate this not by calculating every single indirect reflection (this would be computationally expensive!) but by giving every object in the scene slight illumination with an ambient coefficient term [20].

1.4.2 Lambertian Shading

Shading of diffuse surfaces is also known as *Lambertian* shading, named because Lambert's cosine law is essential for modeling diffuse surfaces. The law states that "the color e of a surface is proportional to the cosine of the angle between the surface normal and the direction of the light source" [20]. The $\cos\theta$ between the light source direction and the surface normal is calculated to determine how much light reaches the surface. As mentioned prior, the microscopic roughness of a diffuse surface can be represented with microfacets. Microfacets at the surface cause the surface normal to vary even at the same point. This means incoming light will scatter wildly, seemingly in all directions, with the same amount of energy.

The light direction vector l and the surface normal n must be normalized vectors,

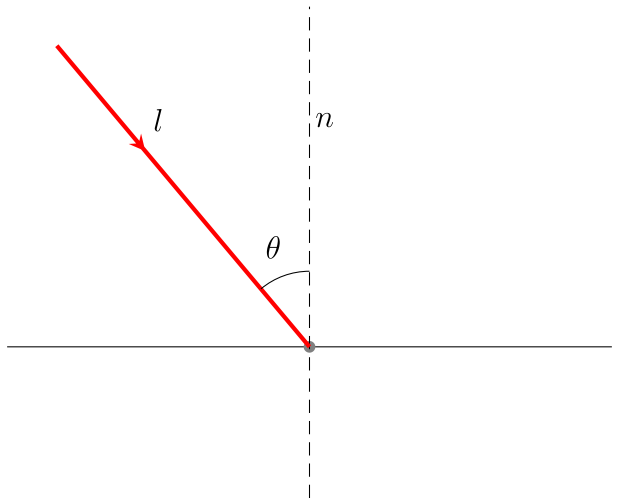


Figure 1.8: The "incoming" light source vector and surface normal.

and we use $l \cdot n$ for $\cos\theta$, since $A \cdot B = |A||B|\cos\theta$ then

$$l \cdot n = \cos\theta$$

Because l and n are normalized, each has a length of 1. The equation for Lambertian shading is as follows:

$$c = c_r c_l (l \cdot n)$$

Where c is the resulting color term, c_r is the color of the diffuse reflectance, and c_l is the color of the light source [20]. Of course, it may seem a little odd to multiply by a color. In reality, these terms, c, c_r, and c_l, are simply storing RGB values as a three element array.

1.4.3 Phong Shading

In 1975, Bui Tuong Phong developed a shading model for specular surfaces [17]. This is known as the Phong Reflectance Model, or the Phong Shading Model[5]. The model relies on an additional eye or "viewing" vector that accounts for the direction of the

[5]Not to be confused with a different shading model he created for smooth shading of surfaces.

eye. Recall that if the viewer is directly in-line with the reflectance vector, the eye will receive maximum brightness. This causes highlights on metal to appear white, when the metal is shone on from a white light source. To model this effect, Phong measured the angle ϕ between the eye and the reflected light. If the angle is 0, the two are the same, and the eye is receiving the reflection at full intensity. An effective

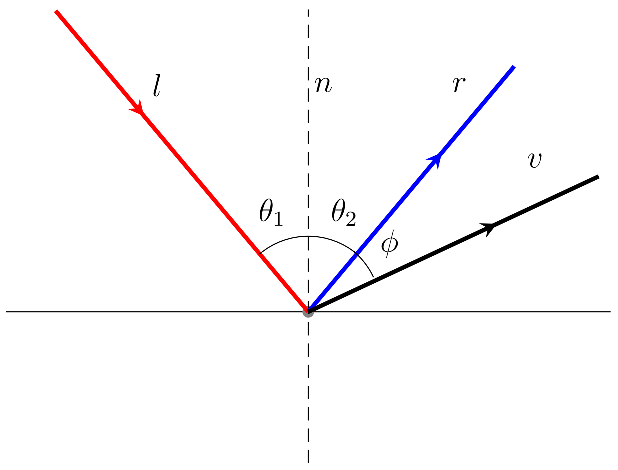


Figure 1.9: The light vector, viewer vector, reflected vector, and surface normal

approximation of how much intensity of light the viewer is receiving, based on the reflection vector, is $\cos\phi$. Because $\cos\phi$ "drops off" too slowly compared to real-life reflections, Phong raised it to the power of p, a rational number known as the *Phong exponent*. To calculate the reflectance vector, we can do the following.

$$b = n(l \cdot n)$$

Which is the projection of l onto n (both are normalized vectors). If l points away from the surface we can "reverse" l by taking the its additive inverse, $-l$, and then add the projection twice to get the reflectance vector.

$$r = -l + 2b$$

Then, to get the $\cos\phi$ between R and V, we can take the dot product of the two vectors, and then raise it to the Phong exponent to get a "faster" fade. The equation for Phong Shading is as follows:

$$c = c_l \max(0, r \cdot v)^p$$

where c is the resulting color, and c_l is the color of the light source [17]. Since the value of $r \cdot v$ can be negative, we clamp the value to 0 to get appropriate values for shading.

Putting it all Together...

Since light can be summed, and most surfaces are a mix of diffuse and specular, we can add the ambient, diffuse, and specular lighting models together in order to get a plausible scene. This is usually implemented with weighted coefficients for the ambient, diffuse, and specular components which are based on the surface type. Using Phong shading for specular surfaces gives us:

$$c = c_r(c_a + c_l \max(0, n \cdot l)) + c_l \max(0, r \cdot v)^p$$

[20].

1.5 Physically-Based Rendering Models

An observant reader may note that the aforementioned models are all approximations of visual effects of light's interactions with surfaces, rather than models for how light actually transports energy (and through that process of transporting energy, transporting color). A model that does accurately portray energy transport is known as a *physically-based* model, as opposed to an empirical or ad hoc model for photorealism.

In order to create a more-accurate model, first we must define what we mean by light transporting energy. Thankfully, the field of radiometry deals with measuring light[6]. Unfortunately, terminology for radiometry is not yet standardized in usage in computer graphics, and so we define terms for use in this paper.

1.5.1 Radiometry Definitions

We begin by defining *flux*, or the radiant energy per time from a light source. This quantity is also known as *power* , and is defined as

$$\Phi = \frac{dQ}{dt}$$

where Q is radiant energy, and Φ is expressed in joules per second (or, more simply, watts). This measures the total energy reaching a surface. However, sometimes energy can be more concentrated in one area, as is the case with glossy reflections. We can express a position-independent quantity, *irradiance* as radiant energy per time per area, defined as

$$E(x) = \frac{d\Phi}{dA}$$

where dA is differential area containing the point x [8]. This quantity is also known as *flux density* in some texts. It is important to keep in mind that irradiance at a point, x, is power *received* at the surface at x.

We can think of radiant emission as "expanding concentric spheres" [11]. When that emission reaches the surface from a light source, we can draw a line the length of the largest sphere's radius from the source to the surface and treat it as a vector. This setup can be seen in Figure 1.10. If a light source is directly above a point, with

[6]The science of photometry also measures light, but only pertains to light within the visible spectrum.

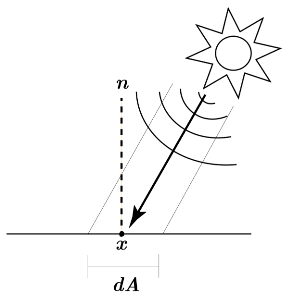


Figure 1.10: A visual of the setup for measuring the energy of light.

its resulting vector in-line with the surface normal, then the irradiance is as follows

$$E(x) = \frac{\frac{dA}{4\pi r^2}\Phi}{dA} = \frac{\Phi}{4\pi r^2}$$

If a light source vector hits the surface at an angle θ (the angle between the normal and the light vector), then foreshortening occurs. This is when In this case we can take the $\cos\theta$ to gauge the area the illumination is spread over. This irradiance formula in this case is then

$$E(x) = \frac{\Phi \cos\theta}{4\pi r^2}$$

What if we want to represent direction? A measure of *radiance* is the amount of energy traveling at some point in a specified direction, per unit time, per unit area, per unit solid angle [21].

This is a bit of a mouthful, and hard to conceptualize. Let's break it down by defining what a solid angle is. A *solid angle* is the analogue of a familiar planar angle, but for spherical geometry, rather than Euclidean. Planar angles are measured in radians, whereas solid angles are in steradians. A solid angle is comparable to taking a slice from a sphere. For example, an entire sphere has a solid angle of 4π, and a hemisphere is 2π [16].

Now that we know what a solid angle is, to put it another way, radiance is flux

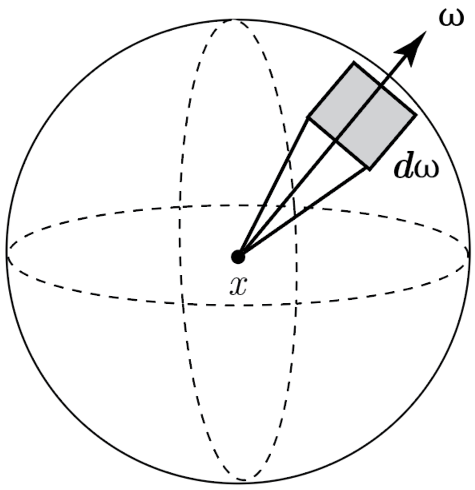


Figure 1.11: An example of the solid angle, represented by the shaded area $d\omega$.

per solid angle [16]. We can write radiance as

$$L(x, \omega_i) = \frac{d\Phi}{d\omega \ dA \cos\theta}$$

where θ is the angle between the normal and light direction vector ω_i, $dA\cos\theta$ is the projected differential area (which accounts for foreshortening), and $d\omega$ is the differential solid angle [11].

1.5.2 Types of Illumination and the Rendering Equation

Direct illumination is computed illumination that only takes illumination from unoccluded light sources into account for a particular point on a surface. The light from the sun when enjoying a lovely spring day is an example of a direct illumination source. This type of illumination does not include reflections of light off of other surfaces, however, so imagine the spring day with harshly shadowed leaves and objects.

Indirect illumination, on the other hand, takes into account incoming light from other surfaces, such as red light reflecting from a red sphere onto white walls. Together, both direct and indirect illumination are used to model a physically plausible scene. By integrating incoming radiance in all directions, we can calculate irradiance as follows

$$E(x) = \int_{\Omega} L_i(x, \omega_i)(n \cdot \omega_i) d\omega \text{ [11]}$$

where we integrate the radiance at point x for each direction ω_i, with $n \cdot \omega_i$ foreshortening, over the the unit hemisphere Ω.

Building off of this idea, in 1986 Kajiya derived a new form of an equation for light transport that maintains equilibrium [9]. This equation is commonly known as the *rendering equation*. Rendering techniques that implement the equation are said to solve the rendering equation. The equation is as follows

$$L_o(x, \omega_o) = L_e(x, \omega_o) + \int_{\Omega} g(x, \omega_i) f_r(x, \omega_i, \omega_o) L_i(x, \omega_i) d\omega_i$$

Overall, the equation captures the radiance at point x outgoing from direction ω_o. The term $L_e(x, \omega_o)$ is the emitted radiance at x via ω_o, and will be greater than 0 only if x is an emitter of light. The next part of the equation is under the integral, meaning all potential incoming light rays at point x will be evaluated via $L_i(x, \omega_i)$. The term $g(x, \omega_i)$ is 0 if x is occluded from light incoming from direction ω_i. Radiance scattered from ω_o at point x, by way of incoming light at direction ω_i is calculated by $f_r(x, \omega_i, \omega_o)$ [9]. This piece of the equation takes in the point x, as well as the incoming light ray ω_i, and gives back the ratio of irradiance from ω_i to radiance emitted from ω_o. This type of function is known as a Bidirectional Reflectance Distribution Function (BRDF).

1.5.3 Bidirectional Reflectance Distribution Function

Bidirectional Reflectance Distribution Functions describe the "ratio of the reflected radiance to incident irradiance" [7], or in other words, outgoing radiance from one direction to incoming irradiance from another direction. The function is defined as follows.

$$f_r(x, \omega_i, w_o) = \frac{dL_o(x, \omega_o)}{dE_i(x)} = \frac{dL_o(x, \omega_o)}{dL_i(x, \omega_i) \cos \theta_i d\omega_i}$$

Its parameters are two direction vectors within the unit hemisphere around point x that represent the direction of incoming light and outgoing light, respectively. These can also be represented as two sets of spherical coordinates, one set for the incoming light vector (ϕ_i, θ_i) and one for the outgoing reflection(ϕ_o, θ_o). Spherical coordinates are in the form (ρ, ϕ, θ) where ρ is the length of the vector from the origin to the point, ϕ is the angle from the horizontal-axis to the point, and θ is the angle from the vertical-axis to the point. Since the incoming and outgoing vectors are normalized, then ρ is implicitly 1. In order to build a truly realistic, physically-accurate model, we must include wavelength as a parameter. It can be explicitly written out as λ, otherwise it is assumed.

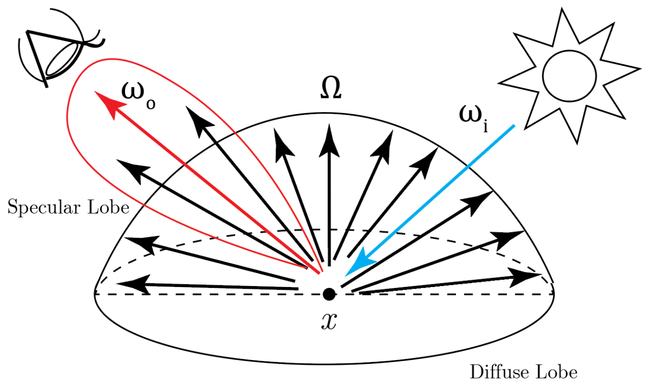


Figure 1.12: A visual of the parameters of a BRDF.

Modern shading models tend to include or modify components of the BRDF to meet their needs. These modifications are usually done for aesthetic or computational purposes. However, BRDFs must retain certain properties to be considered a true BRDF. These properties are

- Bidirectionality, otherwise known as symmetry or reciprocity

$f_r(x, \omega_i, \omega_o, \lambda) = f_r(x, \omega_o, \omega_i, \lambda)$

This is also called Helmholtz reciprocity. [8]

- Energy Conservation: $\int_\Omega f_r(x, \omega_i, \omega_o)(n \cdot \omega_i) d\omega_i \leq 1$

  Outgoing power cannot exceed incoming flux. [11]

One thing to keep in mind is that BRDFs are for reflections. Interactions between transmitted light and surfaces can be represented with a BTDF, a Bidirectional Transmittance Distribution Function. Overall, BRDFs and BTDFs can be described by BSDFs, or Bidirectional Scattering Distribution Functions [7].

Many models have been created with BRDFs, including the Cook-Torrance microfacet model [2], and the Oren-Nayar model for rough, diffuse surfaces [14]. Materials can be measured for their real-world BRDF as well. Models created this way are known as *empirical* models. As before, we can also represent simple reflections off of diffuse and specular surfaces, this time with BRDFs.

Shading with BRDFs

We can represent diffuse, or Lambertian, surfaces with a constant BRDF, $f_r(x, \omega_i, \omega_o) = \frac{P_d}{\pi}$ [11], where P_d is the *albedo*, or the light reflected from the surface. The equation is derived due to the surface properties of a Lambertian surface, as well as the energy conservation requirement of BRDFs [8].

Reflections off a perfectly specular surface, however, are not modeled well using a BRDF, since a BRDF is better suited to show the spread of light. Instead, a Dirac function, δ, is used to model the thin spikes of energy of those surfaces [11]. Of course, most surfaces in the real-world display a combination of diffuse and specular qualities. These are handled by combining the two techniques.

Now that we have a vocabulary to talk about computer graphics concepts, we can put these models of virtual worlds, light, and energy transport into practice and render an image. This can be done with the technique of ray tracing, further discussed in Chapter 2.

Chapter 2

Sketching Light

Ray tracing is one of the major techniques used for rendering computer-generated graphics, the other being rasterization. Although rasterization is the currently popular choice for usage in commercial computer graphics, ray tracing can simulate and produce more realistic images because of its representation of light. This makes it a better choice, along with physically-based shading models, for modeling iridescence. Below, we explore the history of ray tracing, as well as what steps and choices the technique takes to produce a realistic end result. In addition to this, we examine different illumination models and the benefits and drawbacks of each model.

2.1 Motivating Factors for Ray Tracing

Ray tracing, as the name suggests, is the process of "tracing" rays of light in order to generate a 3D image. Light sources such as the sun or light bulbs emit a large quantity of photons, or "bundles of electromagnetic radiation that oscillate with a definite frequency" [12]. Think of a faucet producing a jet of water. Similarly, a light source emits a constant stream of photons of varying energy levels. These photons can reflect or be transmitted when encountering a surface. As previously mentioned, our eyes detect photons when those photons enter the eye and are detected by cones and rods located in the retina. Cones distinguish between different wavelengths, which affects color, while rods distinguish between varying intensities, which affects brightness. We see whatever had lain in the path the photons took from the light source to the eye. Every object "hit" with light that reaches one's eyes is visible—this is the basic principle behind how sight works [7]. In life, this works as follows: Imagine a simple room, illuminated by a single light bulb. Light particles are emitted from the light bulb and bounce around, hitting walls, objects, and the ground before reaching one's eyes. Each photon emitted from the bulb can take a vastly different path from its neighbor, depending on a variety of factors. These include the energy level of the photon, the type of material it encounters, and the starting position of the photon. By seeking to model the path, or ray, each photon takes to the eye, ray tracing effectively aims to model the human visual system in order to produce realistic images.

2.2 Tracing Light Throughout the Ages

It shouldn't come as a surprise that the concept of ray tracing did not originate from the field of computer graphics, although its origins are similarly based in the idea of creating accurate representations of objects. In fact, several fields have used ray tracing principles. In physics, tracing rays of light is hardly a new idea. It is commonly used in the field of optics for the creation of lenses [12]. And in the creation of art, as Steve Luecking points out in his paper, *Dürer, Drawing, and Digital Thinking* [23], 15th century artist Albrecht Dürer used a perspective technique device analogous to digital ray tracing today.

The device consisted of a taut string that served as a ray, attached to a stylus that was run along the contours of the object to be drawn. Dürer would plot points on paper from the position of the string relative to the frame. The resulting images were highly accurate in both proportion and perspective [23].

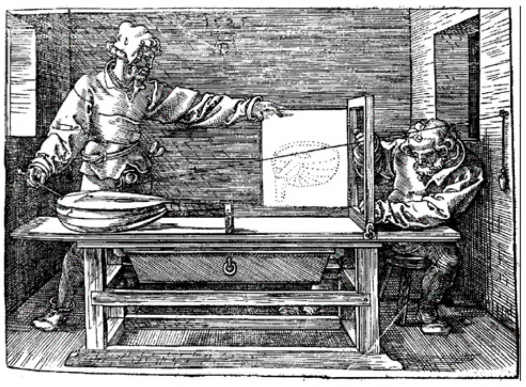


Figure 2.1: Albrecht Dürer's drawing machine.
Taken from Underweysung der Messung mit dem Zirckel und Richtscheyt by Dürer (Public Domain)

As mentioned before, the pinhole camera was another inspiration for ray tracing. The setup of a pinhole camera consists of a closed box with photographic film on the inside of one face, and a pinprick on the opposite face, directly across from the film. Light enters through the pinprick at a certain angle and onto the film, which records

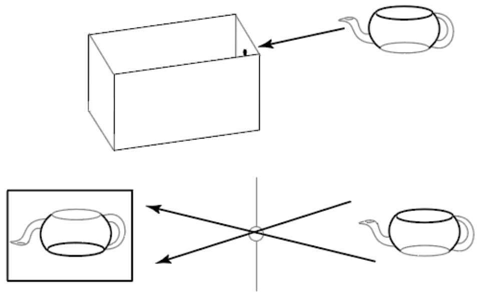


Figure 2.2: A pinhole camera recording a teapot.

where light struck. As seen in Figure 2.2, this process means that light entering the pinhole will strike the film and record objects upside-down. Leaving the pinhole uncovered for longer amounts of time results in more light exposure, subsequently affecting the brightness of the image [6].

2.3 Setup and Implementation

At its most basic, ray tracing implementations in computer graphics are structurally similar to a pinhole camera.

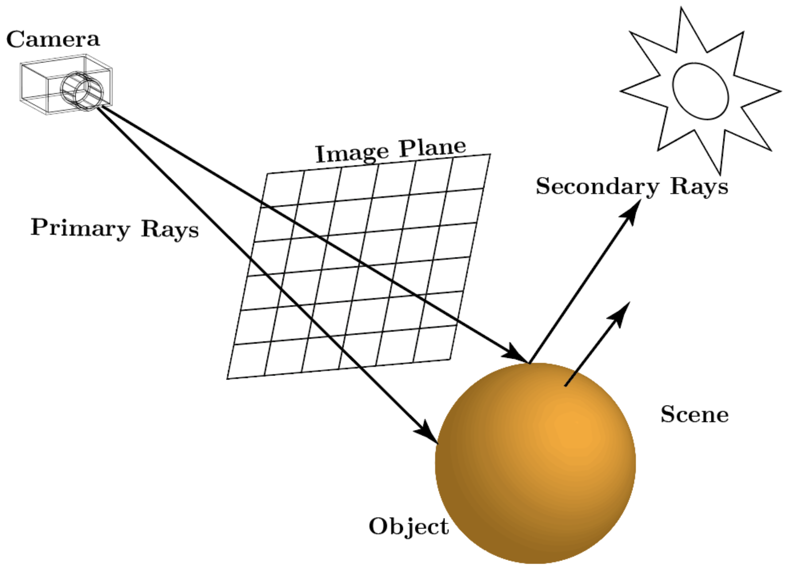


Figure 2.3: A ray tracing setup.

The pinhole is represented by the "eye" or camera of the scene. Recall that images produced by the pinhole camera are upside-down due to light striking the object, entering the pinhole, and then striking the film at the opposite point. To eliminate this problem in computer graphics, the film of a pinhole camera is moved between the eye and the scene, and is called the image plane [6]. This allows for whatever objects are in the scene to directly be recorded onto the image plane via an orthographic or perspective projection. As for modeling light, we could shoot out thousands of rays from the light source and keep track of each one, but this is computationally expensive.[1] Instead, it is common practice to have rays shoot out from the eye/camera, through each pixel on the image plane, and into the scene. This process is known as backward ray tracing [6]. The rays originating from the camera are known as primary rays. If

[1]It can be done, however. This is known as forward ray tracing and is done for some situations alongside other techniques.

a primary ray intersects an object, we note the distance between the origin of the ray and the intersection point, only keeping track of the closest object, as farther ones are occluded.

Algorithm 1: RayTrace(F, loc, img, w, h, scn)

```
// F : focal point of camera
// loc :  a 2D array of size w × h of pixel location.
// img :  a 2D color array of size w × h
// scene :  the scene description
/* Loop through each pixel                                    */
for $j \leftarrow 0$ to $h$ do
    for $i \leftarrow 0$ to $w$ do
        /* We send a ray from the cam to the image plane.     */
        v ← loc [i][j];
        PrimaryRay ← Ray(F, v);
        hit ← None;
        tMin ← ∞;
        foreach object in scn.objects do
            /* We calculate the distance, t of the intersection.  */
            t ← intersect(PrimaryRay, object) // t > 0
            if tMin ≥ t then
                tMin ← t;
                hit ← object;
            end
        end
        if hit is not None then
            P ← F + t v // the intersection point
             view ← (-v / v.length());
            img [ i ][ j ] ← LocalIllum(hit, P, view, scn);
        else
            img [ i ][ j ] ← scn.ambient;
        end
    end
end
```

The pseudo-code above illustrates how image construction and a primary ray intersection would be implemented. The locations of each pixel on the image plane in world space are calculated and stored in *loc*. We loop through each pixel in our $w \times h$

image and create a primary ray that originates from the focal point, F of the camera, and goes through the location of the pixel on the image plane. After that, we loop through each object in the scene and store the closest intersection distance in t, and the closest object in hit. If an object has been hit, we store the point of intersection as P, and then use that point along with information about the scene to calculate the resulting illumination. If an object has not been intersected with, it gets whatever ambient lighting exists in the scene.

After we know what the closest intersection is, the next step is calculating the illumination at that point. To do so we send out a secondary ray from that intersection point to the light source. If the ray reaches the light source, we calculate the pixel color based on the material type, surface normal, color of the material, and color of the light. If it instead intersects with an object, then it is known as a shadow ray, and the point is occluded.

Algorithm 2: Local-Illum(o, P, view, scn)

```
// o :  object to shade
// P : point on object
// view :  direction vector to viewer
// scn :  description of scene
color ← scn.ambient;
// (L, I) = location, intensity of light source
foreach (L,I) in scn.lights do
    l ← (L - P) / (L - P).length() // normalizing light vector
    occluded ← false;
    /* We check if any object occludes the light                */
    foreach object in scn.objects − {o} do
        t ← checkIntersection(l, object)
        if t < (L - P).length() then
            occluded ← true;
        end
        if not occluded then
            /* Calculate the color using a shading model.         */
            color ← color + I * Phong-Color(o, P, l, view);
        end
    end
end
return color
```

Above, we determine whether or not the intersection point, P, receives direct illumination or is occluded. We loop through each light in the scene, each of which is represented by a location and intensity. To check if the object is not illuminated, we do a simple intersection check for l, the normalized vector to the light source, and each object. If any intersection exists before we reach the light source, i.e., if the distance of the intersection is less than the length of l, then the object is occluded. Otherwise, we calculate the color at that point based on a shading model. In this case, we use the Phong shading model to shade the object.

Algorithm 3: Phong-Color(o, P, l, v)

```
// o :  the object being examined
// P : the intersection point on o
// l :  the incident light vector
// v :  the outgoing viewer vector
n ← o.normal(P);
r ← -l + 2(n · l) n;
return (o.diffuse $*$ max(l · n, 0) + o.specular(r · v)$^{o.shininess}$)
```

In the pseudo-code above we calculate the color at point P on object o, using the Phong shading model. We first calculate the surface normal at P, then use that to calculate the reflected ray, r, which is at the same angle as the incident light vector. Then, we calculate the ambient, diffuse, and specular components of the shading model and add them together. The *o.shininess* term is meant as the Phong exponent, as discussed in Chapter 1.

2.4 Illumination Models

Once a ray of light has intersected with an object, we model interactions between that ray and the surface, using different models for different surface types, as previously shown. What is done to calculate the shading depends on the illumination model implemented by the ray tracer. Essentially, once light "hits" the surface of an object, we have options for different models of varying complexities for calculating shading at our disposal. In computer graphics there are two main types of illumination models: local and global illumination.

Local illumination only takes into account unoccluded light sources and the material of the surface. Global illumination recursively calculates light contribution from reflections from other objects in the scene as well as direct light sources to calculate shading. Below, we explore different examples of each type.

2.4.1 Local Illumination

Local illumination models were popular in the 70s and 80s era of computer graphics due to computational limitations at the time. These models are useful for quick calculations and renderings at the expense of depicting realistic effects. They commonly work by taking the viewing ray (from the eye to the intersection point) and performing some calculation with the incoming light ray and the surface normal at the intersection point. Ambient, diffuse, and specular components are calculated separately and added together for most of these models. Examples of local illumina-

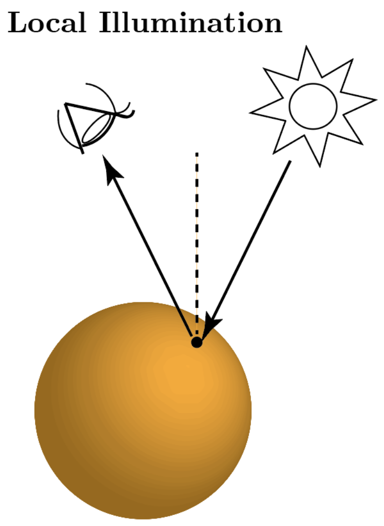


Figure 2.4: An example of local illumination.

tion models include the Lambertian shading and the Phong shading model described earlier.

2.4.2 Global Illumination

Despite the (relatively) quick computations, computer graphics researchers at the time were dissatisfied with the plastic-looking, hard-shadowed images that were being produced by ray tracing with local illumination models. These images were unfaithful to life, to the actual behavior of light photons. In real life, shadows aren't stark black, but exhibit inter-reflected light. The same goes for light reflecting off one surface onto another, which often exhibits color bleed. This is when the color of a surface displays color from light reflecting off of a different surface.

In 1979, Turner Whitted changed the rendering game with his seminal paper, *An Improved Illumination Model for Shaded Display* [25] which devised a way to represent global illumination, to an extent. The paper not only introduced recursively calculating rays (with shadows and color bleed for free), but also introduced refraction and adaptive super-sampling. He noted that one can represent the ways reflections of light off a surface can contribute to other points in a scene as a "tree" of rays. In practice, this can be modeled by shooting off a new reflection ray for every intersection point, as well as a ray to the light source. This was slightly outrageous because of the added computational expense, but highly effective in producing believable lighting effects [25].

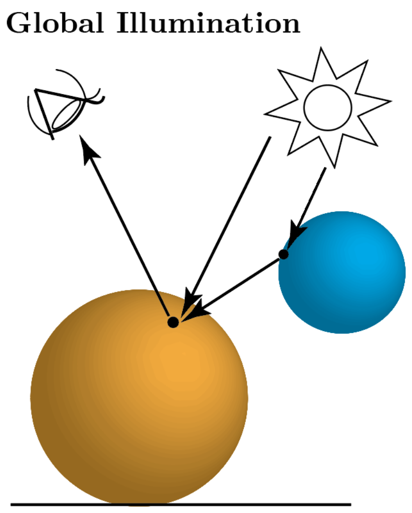


Figure 2.5: An example of global illumination.

However, the Whitted ray tracing model does not require shading models to simulate energy. Researchers then found ways to incorporate shading with the energy distributions from BRDFs into global illumination models.

Another model, one that does take into account energy transfer, is distributed ray tracing, also known as stochastic ray tracing. This was created by Cook, Porter, and Carpenter in 1984 in their paper *Distributed Ray Tracing* [3]. The model also uses supersampling, shooting out numerous rays at a point. For each point, the model stochastically samples new points from the BRDF hemisphere distribution and calculates their contribution to the point's radiance, then does the same process with those sampled points. This effectively simulates photons bouncing around in real-life. More samples can be taken and then averaged in order to achieve realistic images.

Path tracing is another method for global illumination. Created in 1986 by Kajiya[2], this model also uses stochastic methods. In this method, instead of multiple rays being shot out, a single ray is cast into the scene, and from any intersection point, another ray is stochastically chosen. This is done until a light source is reached. Multiple passes can be done per pixel, then averaged, in order to get a plausible scene [9]. Any shading model that provides a distribution of energy at a point can be used.

These common shading and illumination models are well and good, but one may note that they model light using the primary view of light as a photon. In other words, they do not take our favorite phenomenon of iridescence into account. However, there are more modern shading models researchers have developed that build upon the basic skeleton of ray tracing: these incorporate wavelength, global illumination, and BRDFs in order to calculate the spectral color of a point on a surface. In the following chapter, we delve into the science behind iridescence, exploring different computer graphics shading models for wave effects, and finally build our own mathematical model for rendering the phenomenon.

[2]This was introduced in the same paper he wrote for the rendering equation.

Chapter 3

“A Play of Colors”: Shading Iridescence

We begin our foray into modeling iridescence with a few examples of the phenomenon, then a scientific overview of the reasons behind why it occurs, as well as what surface setup is necessary for those occurrences. After that, we look into various shading models devised for iridescence, before creating our own formulation for rendering iridescence.

Iridescence, formally known as goniochromism, can be defined as a phenomenon at which colors on a surface appear to change based on the incoming light angle or viewing angle. Recall from prior discussion that some objects and beings, such as soap bubbles and hummingbirds, display iridescence. As seen in life, there is a variety of such objects with iridescent properties. Why is it that some materials display iridescence, while other materials don’t? To answer this question, we can examine objects such as soap bubbles and hummingbirds on a macro and microscopic level, metaphorically squinting through a microscope to get a sense of the underlying structures.

Visually, even to the naked eye, soap bubbles and hummingbirds are vastly differ-

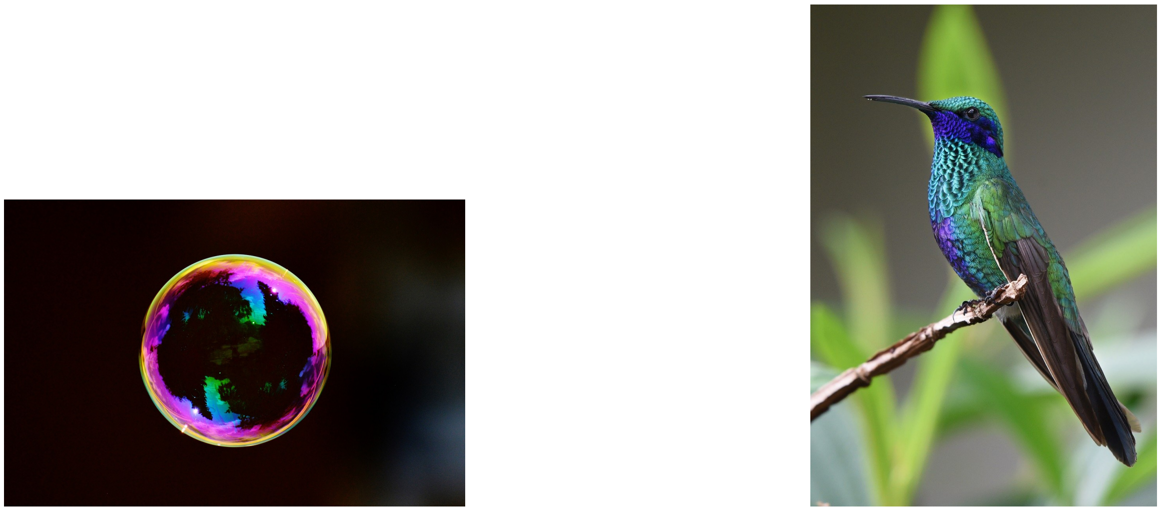

(a) A soap bubble. (b) A hummingbird.

Figure 3.1: A soap bubble and a hummingbird displaying iridescence.
Soap bubble by Alexas_Fotos and Hummingbird by Yan Cabrera (Pixabay.com)

ent. One appears smooth and transparent, displaying a swirl of colors that gradually disappear to white and then black before popping. The other appears soft and downy, catching colors as it darts around. Its vivid feathers are dissimilar to the smooth surface a soap bubble exhibits. Now, let us zoom in on each object, all the way to a microscopic level, on the scale of nanometers. The soap bubble appears similarly to the top layer of a natural pearl, consisting of a microscopic film varying in thickness. Because of gravity, the film at the bottom of the soap bubble will be thicker than the film at the top. In contrast, a feather from a hummingbird now appears like layers and layers of brick, all evenly spaced. Why is it that these structures cause iridescence? Why don't most diffuse or specular surfaces act in the same way? The answer lies in the wavelike properties of light interacting with the surface. Recall that for diffuse and specular surfaces, we can model light *geometrically*, by tracing the path of photons as rays. However, for optical phenomena such as iridescence, light interacts with itself due to its wavelike nature and the surface of the material, canceling out certain wavelengths and amplifying others. Because of these interactions with wavelengths, and the correspondence between wavelength and color, this translates to certain colors being visible depending on the position of the light and the position of the viewer. Below, we focus on the causes of iridescence and then examine ways to model iridescence in computer graphics.

3.1 Light as a Wave

Until this point we have mainly ignored light's wavelike nature[1]. Light waves, which are electromagnetic, can be thought of as carrying energy in an electromagnetic field. A wave has amplitude, wavelength, and frequency. A wave can also be represented by its peaks. These are represented by a series of straight lines called wavefronts. Two noteworthy effects specific to waves are diffraction and interference effects, although there is no clear consensus for distinguishing the two [12]. These two effects due to the wavelike property of light are what cause iridescence. Below, we explore the two effects, examine the surfaces at which they occur, and see how others in computer graphics have modeled them.

Diffraction Effects

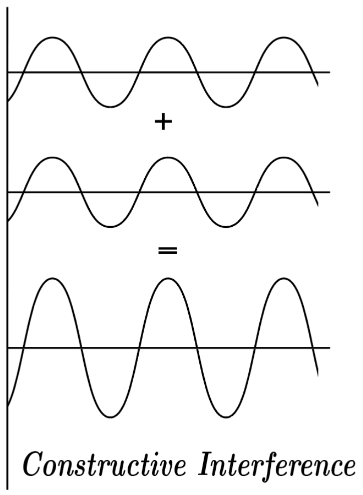
Recall that waves are made up of wavefronts. Usually, these propagate in a straight line. Whenever a wavefront encounters an obstacle or space between solid objects, then the wave will bend around the obstacle. This can lead to interference effects, which are explained in detail below.

Interference Effects

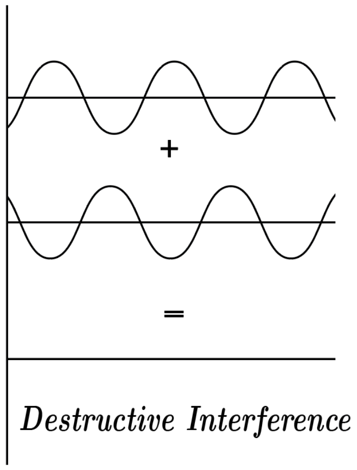
When two waves are *superimposed*, or overlapping, then the amplitude of the resulting wave will be affected. This is known as an interference effect, and can be thought of as combining the amplitude of the two waves at the same points. Because of differences in alignment of the wave form, otherwise known as the phase of the wave,

[1]As do many in the field of computer graphics.

there are two types of interference effects that can occur: constructive and destructive. Constructive interference is when the waves align in wavelength, and combining amplitudes results in an increase in amplitude for the resulting wave.



(a) Constructive Interference



(b) Destructive Interference

Figure 3.2: Constructive and Destructive Interference.

Destructive interference, on the other hand, occurs when the two waves are not aligned in wavelength. That is, one wave has been shifted in wavelength so that peaks do not correspond in peaks in the other wave. This is best illustrated with an example, seen in Figure 3.2. In the figure we can see that amplitude increases for waves that are constructively interfering with each other, while waves that are destructively interfering have decreases in amplitude. Fully destructive interference happens when peaks of one wave hit troughs of another - this occurs when one wave’s wavelength has been π-shifted from another’s. Waves must be *coherent* –have the same frequency and constant phase difference– and, as Marion and Hornyak point out, “interference effects involving light ultimately require that the overlapping waves be generated by the same source” [12] for interference to be visible.

Refraction and Scattering

There are also such colors that rely on refraction rather than pigment or the interference and diffraction effects mentioned above. A popular example is the Tyndall blue of certain birds' feathers, which arises due to light scattering from fine particles in minuscule pockets of air between feathers [26]. These could, perhaps, be thought of as "prismatic" colors.

As Tyndall blue requires light to be scattered from particles in the air in order to be perceived, similarly, interference and diffraction effects have their own conditions for iridescence to occur. These two effects do not cause iridescence in a vacuum, but require the underlying structure of a material to be organized in a such a way that interference or diffraction occurs. This is where the thin films of soap bubbles and the microstructures of natural surfaces come into play. Below, we examine these structures in further detail.

3.2 Microstructures

Microstructures can cause iridescence. As the name suggests, microstructures are surfaces at which on the microscopic level exhibit certain structures or patterns. These structures cause diffraction of light waves as they reflect from the surface, which in turn causes interference effects. These types of surfaces are usually found in nature, although human-made examples are frequently becoming more common today.

Microstructures range from the structure of crystals to the outer shell of a beetle. For example, *Chrysina gloriosa*, a certain green-toned beetle species displaying iridescence, has an outer structure comprised of hexagonal as well as pentagonal cells that react to polarized light [19]. Another example is the sea mouse genus, *Aphrodita*, which possess iridescent spines [15]. Many beetles, birds, snakes, shells, and crystals display iridescence due to microstructures. These microstructures are to light what

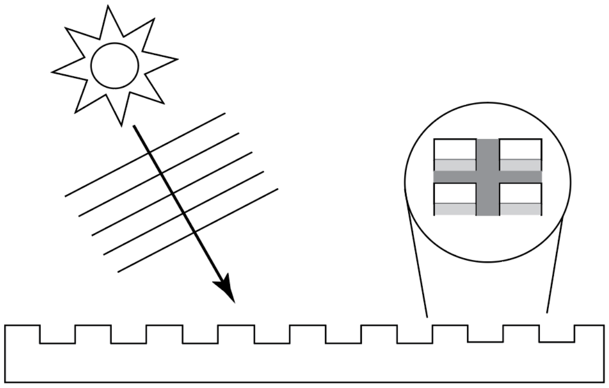


Figure 3.3: A diagram of light hitting a microstructure surface.

logs are to water: diffraction gratings. Diffraction gratings are tools with thin slits used by scientists to cause light wavefronts to split into multiple waves. For example, a diffraction grating could be used to split white light into its spectrum. When light encounters a microstructure, it diffracts, and the resulting waves of light may interfere with each other.

3.3 Thin Film Interference

Iridescence can be caused by interference effects due to light’s interactions with thin films on a surface. As the name suggests, a thin film is simply a minuscule film of a material sandwiched between two other mediums, such as air for a soap bubble. These films are around a thickness of hundreds of nanometers. Some examples of thin films can be seen on soap bubbles, oil slicks, and anti-reflective coatings on the windshields of cars, or cameras. Below, we take a peek below the surface and explore why this structure brings about iridescence.

Thin film interference occurs when a film comprised of one material rests on top of another surface comprised of another material. The thickness of the film must be on the scale of nanometers, close to the wavelength of light entering the medium for interference effects to occur. This setup takes advantage of differing speeds of light in different mediums. When light hits the surface of the film, some of it may reflect off of the surface, and some of it may transmit into the film. That light may refract, depending on the index of refraction of the film. If light is transmitted, when

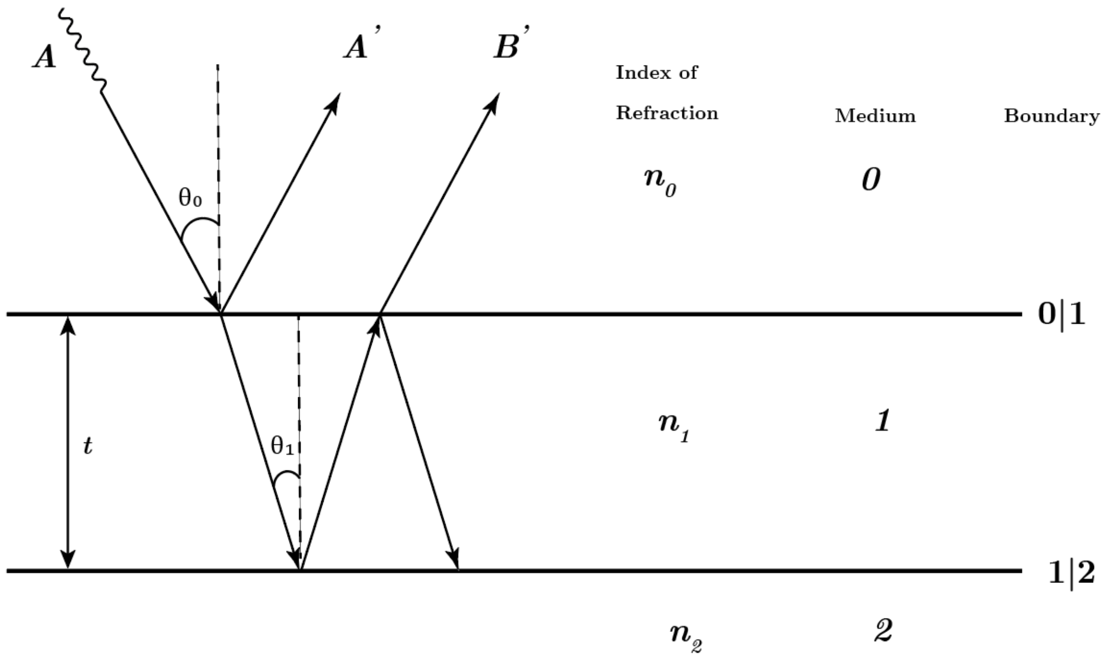


Figure 3.4: A thin film with a light entering.

it reaches the bottom of the film (the interface to the next medium) it may again reflect or transmit. Reflecting light will overlap and either interfere constructively or destructively with the originating light. This is shown in Figure 3.4. If the film is too thick, all waves will interfere constructively. If the film is too thin, destructive interference will take place.

In order to determine whether constructive or destructive interference will occur, we are concerned with the wavelength of the light initially reflected, as well as the wavelength of the light in the thin film. If these are in-phase, constructive interference will occur, usually saturating existing colors or making new ones appear. If the two wave forms are completely out of phase, destructive interference will occur, and the two will cancel each other out. As the phase shift moves from 0 and approaches π, waves will be progressively more destructive. Waves shifted from greater than π to less than 2π will be progressively less destructive. Light reflecting off of a boundary that is of a higher index of refraction will also undergo a π phase shift. As mentioned earlier, if the film of a thin film is too thin, destructive interference will take place. This is due to the π phase shift that will occur due to this type of reflection, as difference in phase due to the path in the film will be negligible.

Different thicknesses of the film will interfere constructively or destructively depending on the wavelength of light, thus producing different colors. Another factor for constructive and destructive interference is the index of refraction of the film. This affects the angle of refraction and subsequently, the length of the path of light in the film, which will affect the phase of the wave.

3.4 Related Work

As previously mentioned, computer graphics tends to ignore wavelike properties of light in favor of the simpler, geometric approach of treating light as a ray. However, researchers have found ways to implement equations for thin films and diffraction effects in the past. In Jos Stam's 1999 paper, *Diffraction Shaders*, Stam renders the diffraction effects visible on CDs. To do so, he extends the He-Torrance shading model using Hemholtz wave equation [22]. More recently, methods for rendering biological diffraction effects due to nanostructures have been created using empirical data [4]. For thin films, Sun and Wang incorporate Fresnel equations for polarization of light for different layering scenarios on thin films [24]. Wu and Zheng turn to microfacet BRDF shading models, creating equations to simulate multi-layer thin films [27]. Barla and Belcour use an analytical approach along with replacing the Fresnel term in microfacet BRDFs with the Airy summation in optics in order to quickly render single thin films [1]. Despite a variety of models on thin films, most of these models tend to be slow and unsuitable for real-time rendering.

Now that we know the science behind these effects and have an idea of what others in the field have focused on, we can calculate and implement equations for iridescence in thin films. Below, I suggest a blend of approximation within a physically-based shading model in order to model thin film interference in single layer films.

3.5 Thin Film Derivation

Our treatment of iridescence focuses on a single-layer thin film setup as seen in Figure 3.5. For simplicity of representation we portray each light wave as a ray. It is important to keep in mind that the rays actually serve as normals to the direction each wavefront is traveling in [12].

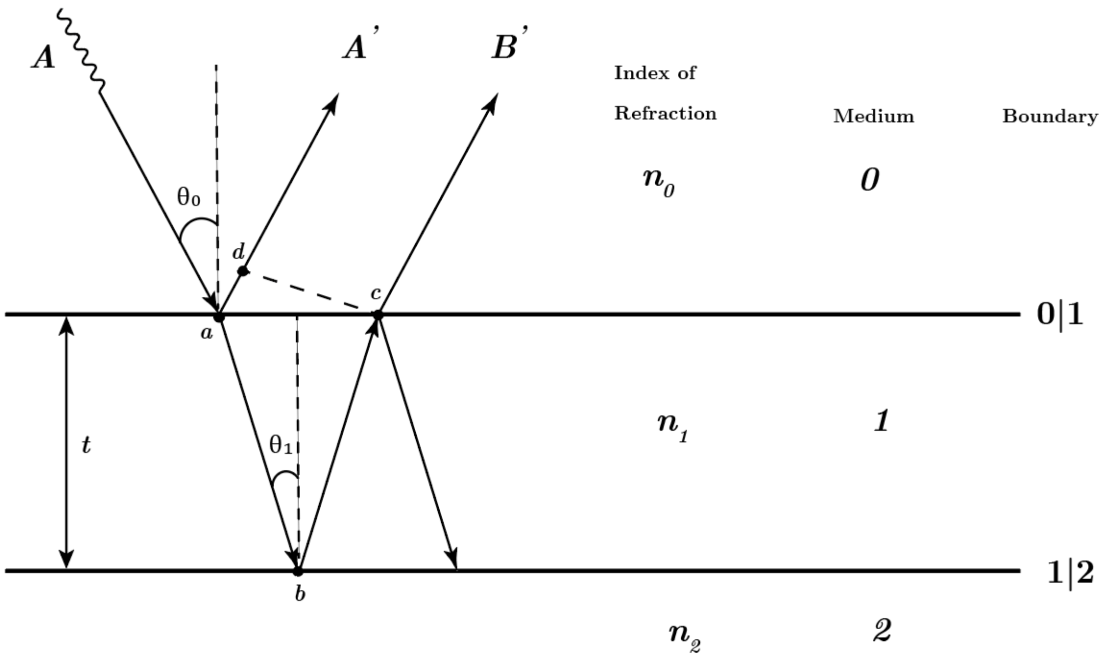


Figure 3.5: A labeled thin film with a light entering.

Our goal is to calculate intensity of the resulting wave, which will be affected due to either constructive or destructive interference. Our film itself is of thickness t, surrounded by an outer medium (0) and an inner medium (2), each with respective indices of refraction. Rays of light interact with the boundary of the outer medium and the film, reflecting away from, and transmitting into, the medium. Those that transmit travel through the film before reaching the boundary of the second medium, and then again reflect and transmit. The refracted rays in the film can reflect and transmit back into the outer medium. In our setup, we ignore absorption that may occur at every interface.

The conditions for interference depend on each phase, ϕ of the two waves. Constructive interference relies on the two waves being an integer number of wavelengths apart, and destructive is anything in between, with total destructive interference requiring the waves to be any integer number of half-wavelengths apart. We can calculate the total phase difference in two parts: we calculate phase shifts that occur due to the path length difference of the two waves, as well as any phase shifts that may have occurred due to the light reflecting off a boundary of a medium with a higher index of refraction. To calculate the difference in path length the waves take, we use what the field of optics calls the *optical path difference* (OPD) to calculate the phase shift. This is the difference in path the two take, multiplied by the respective index of refraction. The length $(\overline{ab} + \overline{bc})$ is the path of the wave in the thin film, while $\overline{ad}$ is the difference in wave path in the outer medium. This difference exists due to the distance where the wave emerges out of the thin film. Since waves can be thought of as orthogonal to the ray, we can shift the ray emerging from the film onto the original reflected ray. This is represented by $\overline{dc}$. The optical path difference then, between the two paths AA' and AB' can be represented as follows.

$$\Delta l = \eta_1(\overline{ab} + \overline{bc}) - \eta_0\overline{ad}$$

Due to the law of reflection $\overline{ab} = \overline{bc}$. The expression then simplifies to

$$\Delta l = 2\eta_1 \overline{ab} - \eta_0 \overline{ad}$$

Which, after a few calculations, further simplifies to

$$\Delta l = 2\eta_1 t \cos\theta$$

since η_0 is assumed to be 1. According to Marion and Hornyak, phase shifts in thin films due to the optical path difference can be calculated by

$$\Delta\phi = 2\pi\Delta l/\lambda \text{ [12]}$$

where we divide the OPD by λ to get the path difference in terms of multiples of the wavelength, and then multiply by 2π to convert to radians. However, this is not the total phase difference between the waves. We must also account for phase differences that occur when waves reflect off a boundary of a medium with a higher index of refraction. This can be represented with

$$b_{ij} = \begin{cases} 0 & \text{if } \eta_j < \eta_i \\ \pi & \text{if } \eta_j > \eta_i \end{cases}$$

which represents the boundary at ij. With this function, we can now add our boundary check function to Marion and Hornyak's equation above to represent the total phase difference as

$$\Delta\phi = (2\pi\Delta l/\lambda) + |(b_{01} - b_{12})|$$

where b_{01} is the reflection from the top boundary, and b_{12} is the reflection from the bottom boundary [12].

Now that we have an equation for $\Delta\phi$, we can calculate the amplitude as follows

$$A' = (A_0 + A_1)\cos(\Delta\phi/2)$$

With this, we can calculate the intensity as

$$I = A^2$$

because intensity is proportional to the amplitude squared [12]. We can then use this intensity equation and calculate the appropriate color for the corresponding pixel.

Below, we present a possible implementation for iridescent shading in pseudocode.

Algorithm 4: Irid-Shading(l, v, o, P)

```
if material == "thin film" then
    Irid-Shading(l, v, o, P);
end
Irid-Shading(l, v, o, P):
// o :  the object being examined
// P : the intersection point on o
// l :  the incident light vector
// v :  the outgoing viewer vector
// (L, I, $\lambda$) = location, intensity, and wavelength of light
// $\lambda$ = [$\lambda_r, \lambda_g, \lambda_b$] (approximating spectrum with RGB channels)
// thin film material has thickness, $\eta_0$, $\eta_1$, $\eta_2$
color;
n $\leftarrow$ o.normal(P);
proj $\leftarrow$ n (l $\cdot$ n) // projection of l onto n
t $\leftarrow$ o.material.thickness;
$\theta_0 \leftarrow \arccos(\text{proj}/\text{l.length}())$;
$\theta_1 \leftarrow \arcsin(\eta_1 \sin\theta_1/\eta_2)$ // Snell's law
foreach channel c in RGB do
    OPD $\leftarrow 2\eta_1 t \cos(\theta_1)$;
    topBoundary $\leftarrow$ (if $\eta_1 > \eta_0$ then $\pi$ else 0);
    bottomBoundary $\leftarrow$ (if $\eta_2 > \eta_1$ then $\pi$ else 0);
    $\Delta\phi \leftarrow 2\pi\text{OPD}/\lambda_c + |(\text{topBoundary} - \text{bottomBoundary})|$;
    A $\leftarrow \sqrt{l.I}$;
    A' $\leftarrow 2A\cos(\Delta\phi/2)$;
    I $\leftarrow A'^2$ ;
    r $\leftarrow$ -l + 2(n $\cdot$ l) n;
    color $_c = \lambda_c *$ I $*$ o.$specular$(r$\cdot$ v)$^{o.shininess}$;
end
return color
```

In this thesis we have presented a model for shading iridescence that is suitable for ray tracing. The model approximates the wave-like nature of light and displays interference effects due to a single-layer thin film. However, the model does not take into account other factors at play when examining light and thin films. Such factors include polarization, which may affect phase shifts, as well as variances in thickness that a real-life surface would exhibit. Also, an approximation model for multi-layer thin films would be a topic for future discussion, as well as the effects of several light sources on thin films.

References

[1] Belcour, L., and Barla, P. A practical extension to microfacet theory for the modeling of varying iridescence. *ACM Transactions on Graphics 36*, 4 (July 2017), 1–14.

[2] Cook, R. L. A Reflectance Model for Computer Graphics. *ACM Transactions on Graphics 1*, 1, 18.

[3] Cook, R. L., Porter, T., and Carpenter, L. Distributed Ray Tracing. Computer Graphics Volume 18, Number 3 July 1984. 9.

[4] Dhillon, D. S., Teyssier, J., Single, M., Gaponenko, I., Milinkovitch, M. C., and Zwicker, M. Interactive Diffraction from Biological Nanostructures. *Computer Graphics Forum 33*, 8 (2014), 177–188.

[5] Dutton, D. *The Art Instinct: Beauty, Pleasure, & Human Evolution.* Oxford University Press, 2009.

[6] Glassner, A. S. *An Introduction to Ray Tracing.* Morgan Kaufmann Publishers, Inc, San Francisco, Calif., 1989.

[7] Glassner, A. S. *Principles of Digital Image Synthesis.* Morgan Kaufmann Series in Computer Graphics and Geometric Modeling. Morgan Kaufmann, San Francisco, 1995.

[8] Hughes, J. F. *Computer Graphics : Principles and Practice*, 3rd ed. Addison-Wesley, Upper Saddle River, New Jersey, 2014.

[9] Kajiya, J. T. The Rendering Equation. 8.

[10] Kolman, B., and Hill, D. *Elementary Linear Algebra.*, 7th ed. Prentice Hall, Upper Saddle River, N.J., 2000.

[11] Kurachi, N. *The Magic of Computer Graphics : Landmarks in Rendering*, english ed. CRC Press, Boca Raton, FL, 2011.

[12] Marion, J., and Hornyak, W. *Physics For Science and Engineering, Part 2*, 1st ed. CBS College Publishing, Maryland, 1982.

[13] Nicolaides, Kimon. *The Natural Way to Draw : A Working Plan for Art Study*. Houghton Mifflin Company, Boston, Massachusetts, 1969.

[14] Oren, M., and Nayar, S. K. Generalization of Lambert's reflectance model. In *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, July 1994), SIGGRAPH '94, Association for Computing Machinery, pp. 239–246.

[15] Parker, A. R., McPhedran, R. C., McKenzie, D. R., Botten, L. C., and Nicorovici, N.-A. P. Aphrodite's iridescence. *Nature 409*, 6816 (Jan. 2001), 36–37.

[16] Pharr, M. Physically based rendering : From theory to implementation, 2004.

[17] Phong, B. T. Illumination for Computer Generated Pictures. 7.

[18] Renner, E. *Pinhole Photography from Historic Technique to Digital Application*, 4th ed. ed. Focal Press, Amsterdam ; Boston, 2008.

[19] Sharma, V., Crne, M., Park, J. O., and Srinivasarao, M. Structural Origin of Circularly Polarized Iridescence in Jeweled Beetles. 4.

[20] Shirley, P. *Fundamentals of Computer Graphics*, 3rd ed. A K Peters, Natick, Mass., 2009.

[21] Sillion, F. X. *Radiosity and Global Illumination.* Morgan Kaufmann Publishers, San Francisco, Calif., 1994.

[22] Stam, J. Diffraction shaders. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques - SIGGRAPH '99* (USA, 1999), ACM Press, pp. 101–110.

[23] Steve Luecking. Durer, Drawing, and Digital Thinking. In *Leonardo's Legacy* (Savannah, Georgia, 2013).

[24] Sun, Y., and Wang, Q. Interference Shaders of Thin Films. *Computer Graphics Forum 27*, 6 (2008), 1607–1631. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1467-8659.2007.01110.x.

[25] Whitted, T. An improved illumination model for shaded display. 7.

[26] Wright, W. D. Iridescence. *Leonardo 7*, 4 (1974), 325–328.

[27] Wu, F.-k., and Zheng, C.-w. Microfacet-based interference simulation for multilayer films. *Graphical Models 78* (Mar. 2015), 26–35.