Ray Tracing in Computer Graphics: Rendering Images to Life

Math 22B Final Project

Jasai Martinez

May 2023

Contents

1 Cover Letter

I decided to continue my project by talking about Computer Graphics from my interest in Math 22A Final Project. I had a ton of fun learning about methods of rotation in computer graphics, specifically about quaternions. I remember in my earlier meetings with Dusty, he also offered ray tracing as another relatable subject connecting to my project about computer graphics.

I started with a broader research proposal where I was going to give generalities about the different ways of applying vector calculus to computer graphics, but the TF reader recommended I narrowed the scope a little bit more towards one technique, which is why I thought ray tracing had been the ultimate perfect way of furthering the project I had started in 22a.

I was also told to start by describing ray tracing, how it works, and some implications and uses of ray tracing. I hope my project aims to answer those questions and teach something new, but I have struggled with the structure of my project and the topics under ray tracing that are most important to cover.

Since a lot of the project touched on computer graphics, my intial draft lacked a lot of visuals. I made sure to include more photos in every section, so everything made a lot more sense. I was also told a lot of formatting was off, so I made sure to cite the photos and where I got them from. I cited definitions and what textbook I got them from. There was also a lack of examples, so I added several throughout every section. I was glad that I stuck to the theme of ray tracing and how rays work in different ways to render images. I hope you enjoy my project!

2 Introduction

This project aims to explore the idea of ray tracing and its importance to computer graphics. In Math 22A, we discovered linear algebra, and in my final project, I covered how objects rotate in computer graphics. Rotations are valuable for showing realistic movements in 3D space. While rotations highlight realism and improve visuals in animation and video games, uncovering how those objects come to life is important.

Ray tracing is a sophisticated technique used in computer graphics to produce lifelike images by replicating the behavior of light when it interacts with objects in a particular setting. Although it is mostly associated with computer graphics, the use of ray tracing extends far beyond this field. Without ray tracing, these graphical representations would not be as precise, particularly with regard to the way light and shadows are presented.

How could images exist and render accurate lighting? What about shadows? What are other methods for rendering lighting and shadows? How do we deal with reflections and refractions?

Through this project, I will answer all these questions and more as we discover the processes of ray tracing and its applications.

3 What is Ray Tracing?

Objects placed in a scene through digital graphics can be broken down into a projection from 2D space. The space in 2D captures the width and height of an image, and the projection of the image leads to a 3D image capturing depth. Each 2D image is broken down into pixels which can be represented as points on an $xy-$plane. Each pixel on the screen corresponds to a specific location in the 2D image, and its color and brightness determine how that location is displayed. Extending a point to a direction on the xy-plane, we can create rays.

Definition 1: A **ray** is a semi-infinite line defined by its origin e and its direction vector $\vec{d}$.[10]

3.1 Ray-Object Intersection

Using Definition 1, we can construct equations for rays and develop the idea behind an intersection between rays and objects. Let's use spheres as an example of ray-object intersection.

Let $p(t) = e + td$ represent a ray in 3D space, where e is the ray's origin, d is the ray's direction, and t is a parameter that controls how far along the ray we are. The function $f(p) = 0$ represents an implicit surface that the ray is intersecting. Here, p is a function of x, y, and z, representing a point in 3D space. The equation $f(p) = 0$ gives us a way to determine if a point p lies on the surface or not. If $f(p) = 0$, then p is on the surface. Otherwise, it is not.

Then let $f(p) = 0$. We can use a sphere with center $c =< x_c, y_c, z_c >$, and the formula represented by our implicit equations is as follows:

$$(x - x_c)^2 + (y - y_c)^2 + (z - z_c)^2 - R^2 = 0 \tag{1}$$

$$= (p - c) \cdot (p - c) - R^2 = 0 \tag{2}$$

Using our second equation, such that p and c represent vectors, we can continue expanding to solve for parameter t in our ray:

$$(e + td - c) \cdot (e + td - c) - R^2 = 0$$

$$= (d \cdot d)t^2 + 2d \cdot (e - c)t + (e - c) \cdot (e - c) - R^2 = 0$$

Note: We see a quadratic equation where we can solve for t

$$t = \frac{-d \cdot (e - c) \pm \sqrt{(d \cdot (e - c))^2 - (d \cdot d)((e - c) \cdot (e - c) - R^2)}}{(d \cdot d)}$$

The discriminant for our solution can provide us with three different solutions and explanations for ray-object intersection with a sphere:
(1) The discriminant is negative: The ray and sphere do not intersect
(2) The discriminant is positive: There are two solutions (ray enters, ray leaves)
(3) The discriminant is zero: The ray grazes the sphere touching at exactly one point

The intersection point in the solution of our parameter t is then used to calculate the color and shading of the sphere, contributing to the final image.

3.2 Ray Tracing Algorithm

The ray-object intersection of a sphere is a fundamental component of the ray tracing algorithm. In the ray tracing algorithm, a ray is projected from the virtual camera and traced as it interacts with objects in the virtual scene.
The ability to accurately calculate ray-object intersections, particularly with simple shapes like spheres, is essential for rendering realistic images using the ray tracing algorithm. The ray tracing algorithm can be broken down into three simple steps: ray generation, ray intersection, and shading. [6]

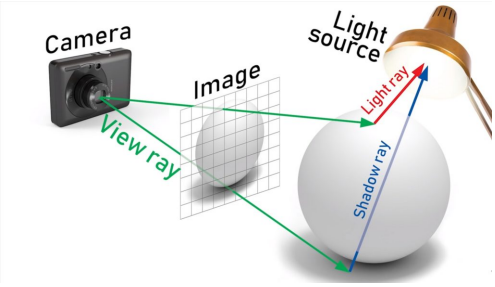

Figure 1: A viewing camera rendering a sphere – showing shadow & light rays bouncing off the sphere [9]

Beginning with *ray generation*, we focus on the geometry of the virtual camera. The camera fixates on the viewing angle and generates rays that shoot onto the 3D scene. Each ray corresponds to a pixel in the final image, and rays are typically generated in a grid pattern to cover the entire image area.

Once rays are created, we test for *intersection* with the objects in the 3D scene. Starting from the closest object to the camera, we follow by computing the intersection of the ray with every object in our 3D scene. The example with spheres from *Section 3.1* can be similarly applied to other mathematical equations for calculating the intersection of a ray and differently shaped objects. If the ray intersects with an object, the point of intersection is recorded, and then we move on to shading.

Once the intersection points are determined, the color and brightness of the pixel corresponding to each ray are calculated through *shading*. We can use the position and color of the light sources in our 3D scene, the angle of intersection relative to the camera, and the object's material properties (reflection and refraction, which will later be discussed). These components are important to determine the color and brightness of each pixel that is added to the final processed image.

Note in Figure 1, through ray tracing, we generate an image using the rays that intersect with the surface and bounce towards the light. However, we must

also account for the rays that do not find a light source, which leads us to generate shading.

4 Shading

Now that we have an algorithm that develops the calculations for ray tracing, we can use those concepts to set the scene further. We have the foundations for ray generation and ray intersection, but let's introduce the third component, shading! If any ray intersects an object on its path to the light source, then we consider this a **shadow ray**.

4.1 Blinn-Phong Shading

To render shading in our images, we can utilize Blinn-Phong shading. Blinn-Phong is a popular illumination model used in computer graphics to simulate the interaction of light with a 3D surface. We can calculate the shading at each point based on the position of the light source, the orientation of our object, and the view of direction.

4.1.1 Blinn-Phong v. Phong Shading

The Blinn-Phong shading model is based on the Phong shading model – used to calculate the color of a point on a surface based on the angle between the surface normal and the direction of the light source. In the Phong model, the specular highlight is calculated by taking the reflection of the incoming light vector around the surface normal, $\vec{R}$, and then computing the dot product of that reflection vector with the view direction, $\vec{V}$: $(R \cdot V)$. This computation involves costly trigonometric functions, which can be slow to evaluate.

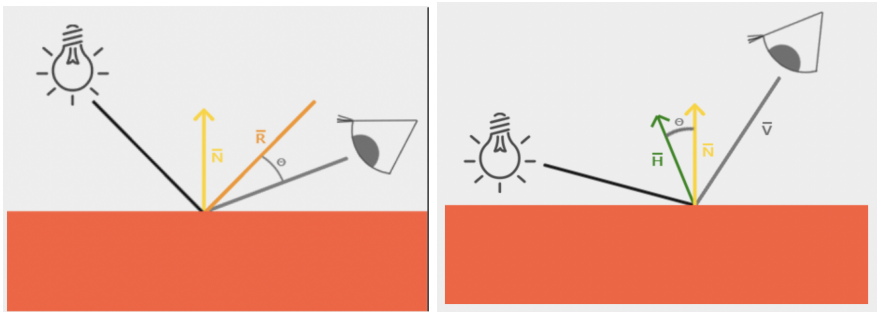

Figure 2: Pictured on the left: Phong Shading, with the reflection vector, $\vec{R}$, and the Viewer direction vector, $\vec{V}$.
Pictured on the right: Blinn-Phong Shading, with the inclusion of the halfway vector, $\vec{H}$.[5]

The Blinn-Phong model simplifies this calculation by using the halfway vector between the incoming light vector and the view direction as the basis for the specular highlight computation and taking the dot project with the normalized surface. We calculate Blinn-Phong shading as $\vec{H} \cdot \vec{N}$.

Definition 2: The **halfway vector**, $\vec{H}$, is the vector that lies midway between the direction vector of a light source, $\vec{L}$, and the viewer's eye direction vector, $\vec{V}$, at a given surface point. We define $\vec{H}$ as:

$$\vec{H} = \frac{\vec{V} + \vec{L}}{\|\vec{V} + \vec{L}\|}[5]$$

This computation is faster than the Phong model since it only involves one vector addition and one normalization, rather than a reflection computation and a dot product with a reflection vector.

While this may seem like a simple calculation, this process of Blinn-Phong shading helps to create the appearance of highlights on the surface when light is reflected at certain angles. Let's take a look at an example, continuing our usage of the sphere.

4.1.2 Blinn-Phong Shading Example

Suppose we have a 3D scene with a sphere at the origin and a point, L, which represents the light source at position (1, 1, 1). We want to calculate the shading at point P on the surface of the sphere, given that the surface normal at that point is N and the view direction is V.

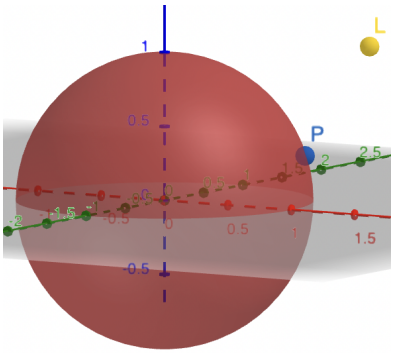

Figure 3: Sphere with radius 1 at the origin, and point$P$ on the sphere, and light source L [1]

To apply our Blinn-Phong shading method, we calculate the halfway vector $\vec{H}$ between the incoming light vector $\vec{L}$ (the vector from the point P to the light source) and the view direction V. This is given by: $\vec{H} = \frac{\vec{V}+\vec{L}}{\|\vec{V}+\vec{L}\|}$ Next, we

calculate the dot product of the surface normal N and the halfway vector H, $\vec{N} \cdot \vec{H}$.

The dot product of the normalized surface normal vector and the normalized halfway vector gives us a scalar value that determines the intensity of the specular highlight. The range of this scalar value is between -1 and 1 because the cosine of the angle between two vectors can range from -1 to 1 (since the dot product is the product of the magnitudes of the two vectors, the resulting value will be between -1 and 1).

Note: The specular highlight and full calculations of Blinn-PhonG shading require the application of more complex scenes and the specular component, so the exact result depends on several factors (i.e, the viewing direction), but let's understand its implications!

If this value is close to 1, it means the surface is facing directly toward the light source and should therefore be more reflective, resulting in a brighter shading. If it's close to -1, it means the surface is facing away from the light source and should be less reflective, resulting in darker shading. If it's 0, it means the surface is facing sideways relative to the light source and should have intermediate reflectivity.

The Blinn-Phong shading model takes into account the positions of the light sources in the scene, the angle of intersection relative to the camera, and the objects that were intersected to determine the final shading of the scene. To fully grasp the complexity of shading, we must also consider reflections in ray tracing.

5 Reflection

Shading techniques like Blinn-Phong are important for creating realistic 3D graphics, but the appearance of these graphics can be further enhanced through the use of textures, which introduce patterns and variations in the surface properties of objects, such as reflections. Reflection is an important aspect of transporting light, contributing to the realism of rendered images. In ray tracing, we can simulate reflection by tracing additional rays from each surface point to the next reflective surface in the scene. To calculate the direction of the reflected ray, we can use the law of reflection and the reflection vector.

5.1 Law of Reflection

The law of reflection is a mathematical concept that plays a critical role in this story of computer graphics, particularly in the simulation of reflection in ray tracing. This law affects light's behavior when it interacts with a reflective surface.

The incident ray, which is the incoming ray of light, strikes the surface and is reflected back as the reflected ray. The surface normal is the perpendicular line to the surface at the point of incidence, and the angle of incidence, θ_i, is the angle between the incident ray and the surface normal. By knowing the

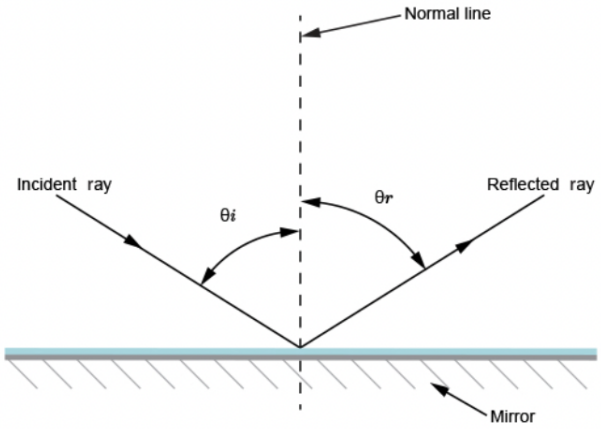

Figure 4: The incident ray (incoming light) bouncing off a mirror and reflecting back a reflected ray[8]

angle of incidence and the surface normal at the point of incidence, the angle of reflection, θ_r, can be calculated using the law of reflection. This information is then used to determine the direction of the reflected ray in the simulation.

Definition 3: The **Law of Reflection** states that the angle of reflection, θ_r, is equal to the angle of incidence, θ_i, for all wave lengths of any material. [12]

The law of reflection states that the angle of incidence equals the angle of reflection. This means that the incident ray and the reflected ray lie on opposite sides of the surface normal, and their angles with respect to the normal are equal. Knowing the direction of the reflected ray is important in ray tracing because it allows us to calculate the lighting and shading effects on the reflective surface. In the case of reflected surfaces in ray tracing, the reflected ray carries information about the direction and intensity of the incoming light, and how it interacts with the surface.

5.2 Reflection Vector

We can calculate the reflected ray by utilizing the reflection vector! The reflection vector is a vector that represents the direction of a reflected ray, calculated using the incident ray and the surface normal at the point of reflection.

Definition 4: A **Reflection Vector** is a vector in the direction of a reflected ray of light off a surface, denoted as: $\vec{r} = \vec{i} - 2(\vec{i} \cdot \vec{n})\vec{n}$, where $\vec{i}$, is the incident ray, and $\vec{n}$, is the normal vector.[7]

By calculating the reflection vector, we can simulate the effect of light bouncing off surfaces in a 3D scene and producing reflections. The reflection vector is applied in the law of reflection, and in combination, both are important to create realistic images in computer graphics.

5.2.1 Example: Applying the Law of Reflection and the Reflection Vector

Let's say we have an incident ray of light with direction vector $\vec{i} = \begin{pmatrix} 1 & 1 & 0 \end{pmatrix}$ that strikes a surface with surface normal vector $\vec{n} = \begin{pmatrix} 0 & 0 & 1 \end{pmatrix}$ at point P. We can use the law of reflection to calculate the direction of the reflected ray by first finding the reflection vector $\vec{r}$.

$$\vec{r} = \vec{i} - 2(\vec{i} \cdot \vec{n})\vec{n}$$

$$\vec{r} = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}.$$

This reflection vector $\vec{r}$ tells us the direction in which the light is reflected off the surface given the incident ray and normal vector. We can utilize the law of reflection and the angle of incidence and reflection in the applications of refractions (Section 6).

5.3 Fresnel Equations

Now that we have discussed the basics of reflection in ray tracing, we can dive deeper into the topic by exploring the Fresnel equations, which are crucial in simulating the reflection and transmission of light at a boundary between two materials in computer graphics.

The Fresnel equations calculate the reflectance and transmittance of light reflected. This fraction of light is a function of the refractive indices of the two materials and the incident angle. Refractive indices are a measure of how much light is bent when it passes through an object. The refractive index of an object determines how much the direction of light is bent when it passes from one medium to another with a different refractive index. This difference in refractive indices between two materials plays a crucial role in determining the amount of reflection and refraction that occurs at their interface.

There are two polarization states that light can have: S-polarization and P-polarization. S-polarized light has its electric field oscillating perpendicular to the plane of incidence (the plane formed by the incident ray and the normal to the surface), while P-polarized light has its electric field oscillating parallel to the plane of incidence.

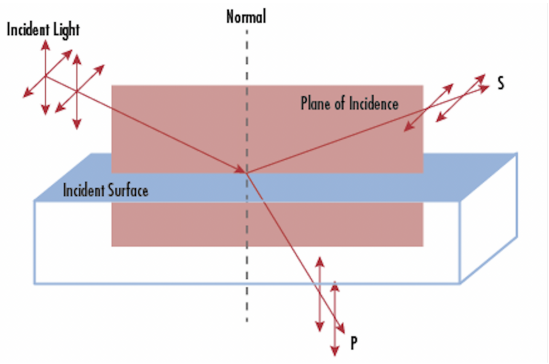

Figure 5: A surface with incident light ray bouncing S-polarized and P-polarized light [11]

Definition 5: The **Fresnel Equations** applied for reflection are as follows:

$$r_p = \frac{n_2 cos\theta_i - n_1 cos\theta_t}{n_2 cos\theta_i + n_1 cos\theta_t} \tag{3}$$

$$R_p = r_p^2 \tag{4}$$

$$r_s = \frac{n_1 cos\theta_i - n_2 cos\theta_t}{n_1 cos\theta_i + n_2 cos\theta_t} \tag{5}$$

$$R_s = r_s^2 \tag{6}$$

$$[3] \tag{7}$$

These four equations calculate the amplitude (R_p and R_s) and reflection coefficients (r_p and r_s) for S-polarized and R-polarized light, contributing to the complexity of capturing light in the process of rendering images.

6 Refraction

Fresnel Equations are also applied to another component of ray tracing in computer graphics. Refraction occurs when light passes through a surface with a different refractive index, such as from air to glass or water. In computer graphics, simulating refraction involves tracing additional rays from each surface point to the next refractive surface in the scene and calculating the direction of the refracted ray using Snell's law.

6.1 Snell's Law

Snell's law is a geometric rule that describes the relationship between the angle of incidence, angle of refraction, and refractive indices of two materials.

It states that the ratio of the sine of the angle of incidence to the sine of the angle of refraction is equal to the ratio of the refractive indices of the materials.

Definition 6: Snell's Law, also known as the law of refraction, describes how light rays change direction when passing through a surface between two different materials:

$$n_1 sin(\theta_1) = n_2 sin(\theta_2) \text{ [12]}$$

Where n_1 and n_2 are the refractive indices of the materials, θ_1 is the angle of incidence, and θ_2 is the angle of refraction.

Once we calculate the angle of refraction, we can use this to determine the direction of the refracted ray. This allows us to accurately predict the behavior of light as it passes through transparent materials. Snell's Law is also applied in the Total Internal Reflection.

6.2 Total Internal Reflection

After calculating the direction of the refracted ray using Snell's law, it is important to consider the possibility of total internal reflection in transparent materials. Total internal reflection occurs when the angle of incidence is greater than the critical angle, and all of the light is reflected back into the same space rather than refracted. This phenomenon is particularly relevant in rendering scenes that include transparent objects, as it can significantly affect the appearance of the object and its surrounding environment.

In ray tracing, we can simulate Total internal reflection by checking if the angle of incidence exceeds the critical angle and then determining the direction of the reflected ray using the Fresnel equations.

Definition 6: The **critical angle** is given by:

$$\theta_{crit} = sin^{-1}(\frac{n_2}{n_1})$$

where n_1 and n_2 are the refractive indices of the materials. [12]

The critical angle is important in total internal reflection because it defines the angle of incidence at which light can no longer pass through a boundary and is instead reflected back into the original medium. Applying the context of ray tracing, it is important to accurately simulate the behavior of light at boundaries between materials with different refractive indices.

Take Figure 6 to highlight the different outcomes dependent on the critical angle's relationship to the incident angle. Suppose the critical angle is 45°. Then there are three possible outcomes:

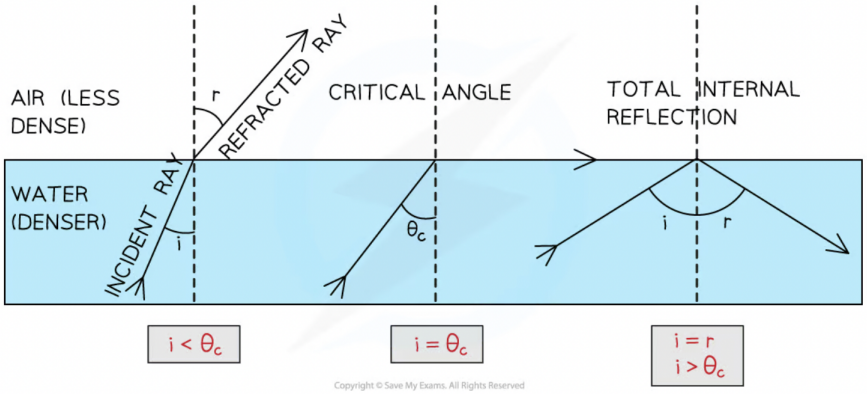

Figure 6: Different examples of incident rays with refractive objects, like water, and the effects of the refracted ray depending on the critical angle [2]

1. The light is refracted and partially reflected if the incident angle is less than the critical angle. When light enters the water at an angle less than the critical angle of 45°, some is refracted, and some are reflected back into the air.
2. If the incident angle is equal to the critical angle, the light is refracted at an angle of 90 degrees and follows the boundary between the water and air.
3. If the incident angle exceeds the critical angle, the light is internally reflected. Thus, when light enters the water at an angle greater than the critical angle, it is totally internally reflected back into the water.

In the greater context of ray tracing, the critical angle is important because it determines whether or not total internal reflection occurs. This is critical for simulating the behavior of light in transparent materials like glass and water, and for creating realistic reflections and refractions in computer graphics.

7 Conclusion: Putting it all together

Now we have an understanding of the most essential components to ray tracing in computer graphics! The importance of ray tracing to produce lifelike images by simulating the behavior of light when it interacts with objects in a particular setting is used in most modern graphics.

In this project, we have discussed the various aspects of ray tracing, including the ray-object intersection, the ray tracing algorithm, and shading techniques such as Blinn-Phong shading. We have also explored reflection, including the law of reflection and the reflection vector, as well as refraction, including Snell's law and total internal reflection. The importance of ray tracing in rendering precise and accurate images with accurate lighting and shadows is throughout our lives in modern technology! Continue using technology and be on the lookout for how you view light and its contribution to bringing images to life.

References

[1] Geogebra 3d. https://www.geogebra.org/3d?lang=en.

[2] Total internal reflection. https://www.savemyexams.co.uk/igcse/physics/edexcel/19/revision-notes/3-waves/3-2-reflection–refraction/3-2-6-total-internal-reflection/.

[3] James D Foley, Foley Dan Van, Andries Van Dam, Steven K Feiner, and John F Hughes. *Computer graphics: principles and practice*, volume 12110. Addison-Wesley Professional, 1996.

[4] Andrew S Glassner. *An introduction to ray tracing*. Morgan Kaufmann, 1989.

[5] LearnOpenGL. Advanced lighting, n.d.

[6] Steve Marschner and Peter Shirley. *Fundamentals of computer graphics*. CRC Press, 2018.

[7] Wojciech Matusik. Ray tracing, 2012.

[8] Mammoth Memory. The three laws of reflection. https://mammothmemory.net/physics/mirrors/flat-mirrors/the-three-laws-of-reflection.html.

[9] Sydney Mugerwa, George Kimathi, and Odunayo Ezekiel. What is ray tracing technology and how it works in gpus, Sep 2022.

[10] Matt Pharr, Wenzel Jakob, and Greg Humphreys. *Physically based rendering: From theory to implementation*. Morgan Kaufmann, 2016.

[11] Edmund Optics Worldwide. Introduction to polarization. https://www.edmundoptics.com/knowledge-center/application-notes/optics/introduction-to-polarization/, n.d.

[12] Hugh D Young and Roger A Freedman. *University physics*, volume 9.