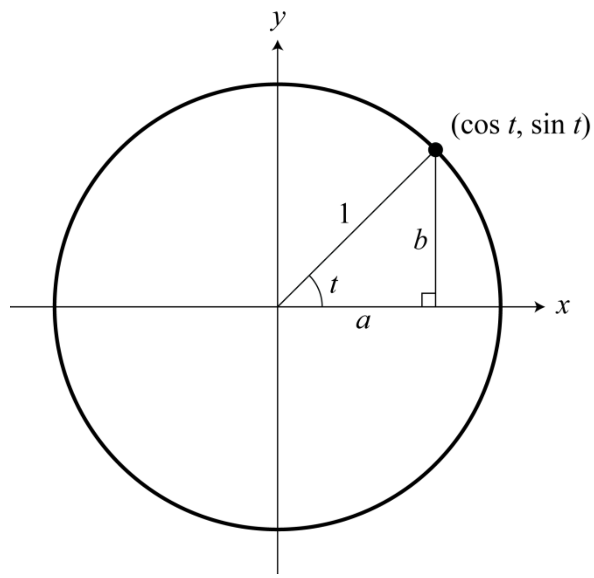
Bi-Directional Reflectance Distribution Functions in Computer Graphics

by Isaias Sifuentes



The unit circle, unit sphere, and trigonometry all bear a direct relationship, and both have countless uses in the field of computer graphics, especially with respect to Bidirectional Reflectance Distribution Functions (BRDFs) and BRDF-based lighting techniques. Consider the Pythagorean Theorem:

$$a^2 + b^2 = c^2$$

The unit circle is a circle with a radius of one in the Cartesian coordinate system, centered at the origin (0, 0). Coincidentally, the Pythagorean Theorem holds true for all right triangles derived from all points *(x, y)* on the unit circle.

Furthermore, consider a ray starting from the origin to a point *(x, y)* on the unit circle that makes an angle *t*. This would form a right triangle with legs *a* and *b*, and a hypotenuse of one. Then we may also describe the point *(x, y)* as:

$$(\cos t\,, \sin t)$$

We may verify these relationships as follows. Consider the following isosceles triangle from the diagram:

$$t = 45° \; and \; b = a$$
$$a^2 + a^2 = c^2$$
$$2a^2 = 1$$
$$a^2 = \frac{1}{2}$$
$$a = \sqrt{\frac{1}{2}}$$

The trigonometric ratios of sine, cosine, and tangent can be used to find unknown angles and lengths of unknown sides. The cosine of an angle is the ratio of the length of the adjacent side to the length of the hypotenuse. Therefore:

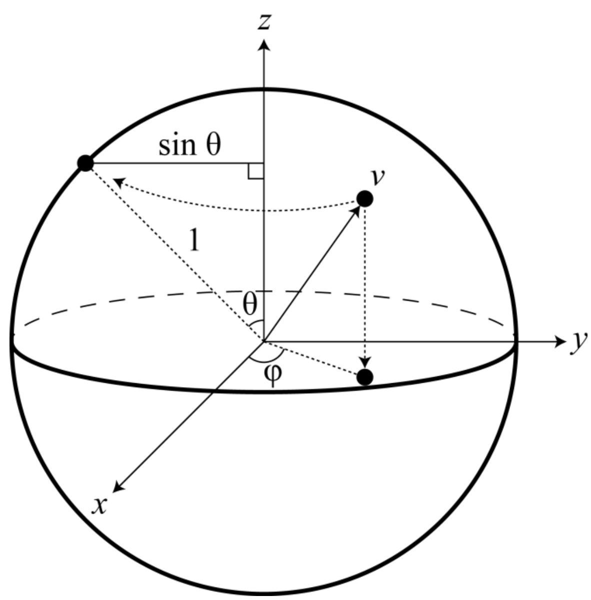
$$\cos(t) = \frac{\sqrt{\frac{1}{2}}}{1} = \sqrt{\frac{1}{2}} = \frac{\sqrt{1}}{\sqrt{2}} = \frac{1}{\sqrt{2}} \cdot \frac{\sqrt{2}}{\sqrt{2}} = \frac{\sqrt{2}}{2}$$

Or, more simply:

$\cos t = \frac{x}{1} = x$	$\cos 0° = 1$	$\cos 90° = 0$
$\sin t = \frac{y}{1} = y$	$\sin 0° = 0$	$\sin 90° = 1$

Let us now turn our attention to the unit sphere. The unit sphere is nearly the same as the unit circle. It still has a radius of one, but has a third additional dimension.

While Cartesian-space vector notation of the form $v = (v_x, v_y, v_z)$ is useful for performing many types of computations, it is cumbersome when trying to parameterize BRDFs. Instead, it is more useful to represent a vector in spherical coordinates by a magnitude ρ and a pair of angles θ and φ.



If we rotate the vector *v* along the z-axis to be in the z-y plane, we can use trigonometric ratios to determine that the point is some quantity $\sin\theta$ away from the z-axis and some quantity $\cos\theta$ above the x-y plane; therefore $v_z = \cos\theta$.

The angle φ is obtained by modifying *v* such that $v = (v_x, v_y, 0)$. With the point now lying in the x-y plane and forming a right-triangle with a hypotenuse equal to $\sin\theta$, trigonometric ratios are used once more to determine that:

$$v_x = \cos\varphi \sin\theta \,; v_y = \sin\varphi \sin\theta \,;\; v_z = \cos\theta$$

The aforementioned formulas are used when converting spherical coordinates to Cartesian coordinates; only the angles θ and φ are needed for this process. The formulas for converting Cartesian coordinates into spherical coordinates are:

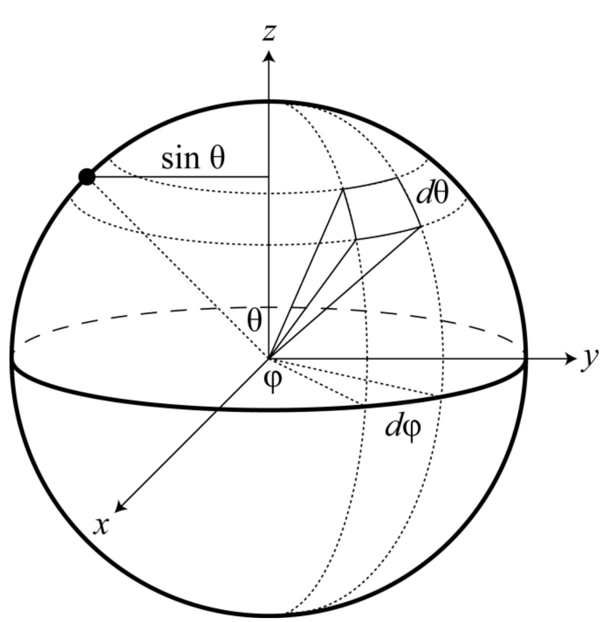
$$\tan\varphi = \frac{v_y}{v_x}$$

$$\varphi = \arctan\frac{v_y}{v_x}$$

Since $v_z = \cos\theta$,

$$\theta = \arccos v_z$$

We are now able to move on to a formal definition of the BRDF, bearing in mind some terminology. The angle φ is sometimes referred to as the azimuth angle, while the angle θ is sometimes referred to as the polar angle or zenith angle. The polar angle is measured from a fixed zenith direction, namely the z-axis. The azimuthal angle of a point consists of its orthogonal projection onto the x-y plane, and the zenith direction is orthogonal to this x-y plane.



Much of the illumination in the real world happens in terms of area lighting, so it follows that we must build illumination models accordingly. In this model, since incident and reflected light occur over an area, we must introduce the differential solid angle. In the illustration, the differential solid angle is the area on the unit sphere that is intersected by the volume of directions. Given small angular changes $d\theta$ and $d\varphi$, the differential solid angle is defined as:

$$d\omega = (height)(width)$$
$$d\omega = (d\theta)(\sin\theta \, d\varphi)$$
$$d\omega = \sin\theta \, d\theta d\varphi$$

It is important to remember that the width of the differential solid angle will be modified by a value of $\sin\theta$, and this makes sense both geometrically and numerically. For values close to one, $\sin\theta$ will be very close to the azimuth and almost exactly as wide as $d\varphi$. However, for values close to zero, $\sin\theta$ will be very close to the zenith and will cause $d\varphi$ to be nearly zero. Consequently, the width of the area changes by a factor of $\sin\theta$.

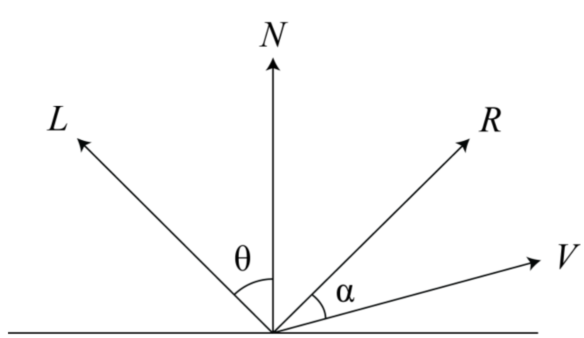
There are also two kinds of material surfaces one must bear in mind with regards to BRDFs: isotropic and anisotropic. Isotropic surfaces are surfaces whose BRDFs do not change when rotated about their normal. Materials like polished chrome and smooth plastic are isotropic surfaces. Anisotropic surfaces, however, have reflectance properties that exhibit change when these surfaces are rotated about their normal. Brushed metal, satin, and hair are anisotropic surfaces.

The BRDF is defined as a ratio of radiance to irradiance:

$$BRDF = \frac{L_o}{E_i}$$

where L_o is equal to the radiance and E_i is equal to the irradiance. It plays an important role in modern computer graphics, as many photorealistic rendering algorithms use the BRDF as their foundation. Thus far, we have seen a brief introduction to BRDFs and the optical physics that define them. Let us now proceed to examine how the BRDF is processed in the world of computer graphics.

A typical software implementation of computer graphics illumination models consists of “shaders”, which may be written for a number of various platforms. One of the most popular rendering applications called Mental Ray supports shaders written in the C or C++ programming languages. Pixar Animation Studios currently develops and maintains PhotoRealistic RenderMan, a RenderMan-compliant renderer for which shaders can be written in the RenderMan Shading Language. Yet another shading language is the OpenGL Shading Language, created by The OpenGL Architecture Review Board. Although each shading language has its own API, their syntax tends to be C-like.



Much of the code in shaders is a software implementation of techniques from the mathematics field of Linear Algebra. Shader APIs have built-in functions to calculate typical vector operations like addition, subtraction, multiplication, dot products, and cross products. The vectors that are most commonly referenced in both illumination models and their shader counterparts are the light vector L, the surface normal N, the light reflectance R, and the camera or view vector V.

Although the BRDF has allowed computer graphics artists to achieve impressive results, it is only a starting point. It is a small analysis of light phenomena but is hardly comprehensive due to many factors, especially wave and particle theories of light. Numerically, these illumination models employ Integral Calculus over a hemisphere local to a point on the surface of geometry. However, these operations are too intensive and expensive for modern computing hardware. Instead, a small amount of individual point light sources are sampled to compute the illumination of a surface for an acceptable approximation.

Other modern illumination models extend the BRDF well beyond its original capacity to emulate light phenomena more accurately. In order to produce convincing objects and materials like water, glass, skin, wax, milk, marble, and jade, computer graphics research must take into consideration the wave theory of light. For this reason the Fresnel equation, which describes light behavior when passing through materials with different refractive indices like water, glass, and skin, is a component of some models.

Convincing human models and skin, for example, require a combination of a sufficient subsurface scattering algorithm along with high-definition textures for the model. At the very least, a subsurface scattering shader along with a bump map is enough to produce a rough draft of human skin. Real-time rendering speeds of these algorithms in video games are another force driving their efficiency. Subsurface-scattering research for computer graphics currently indicates more than one algorithm to choose from depending on the need for speed.

In conclusion, photorealistic computer graphics is an area of active research. There are many observable light phenomena which have yet to be expressed as illumination models that can be easily digested by a computer. Commonly, a device called a gonioreflectometer has been used to measure BRDFs for certain objects. The measured BRDFs are then compared to the results of traditional illumination models to test for accuracy and areas for improvement. Some experts question whether computer graphics will ever reach a point beyond the uncanny valley and whether true photorealism can ever be attained, at least in interactive environments.