# Chapter 19

# Color

In this section, we will explore the basic concept of color. We will talk about what color is, and various ways to represent it. This is a rich topic of study, and many mysteries about human color perception remain unanswered. We will spend extra time on this subject, as we find it very interesting, and due to its importance not just to computer graphics but to digital imaging as well.

Color is, in fact, an overloaded term meaning many different things. When a light beam hits the retina, there is some initial neural response by the *cone-cells* that occurs independently at each cell. We can refer to this as *retinal color*. Retinal color is then processed in an integrated fashion over the entire field of view resulting in the *perceived color* that we actually experience and base judgments upon. The perceived color is often associated with the object we are observing, which we might call the *object color*.

At all of these stages, the simplest thing we can say about two particular colors is simply whether they are the same or different. This is something that we can often record and quantify, and it is the main way we will deal with color in this chapter. At the perceptual color level, there is clearly a conscious experience of color which is much harder to deal with experimentally or formally.

Finally, there are further issues in how we typically organize our color perceptions into *named colors*, using words like red and green.

In this chapter, we will mostly focus on retinal color (and will drop the term retinal). Retinal color theory is relatively well understood, and is the first step in understanding other notions of color. We will first describe retinal color from its, now, well established bio-physical basis. Then, we will re-derive the same model directly from perceptual experiments. We will then discuss some of the most common color representations in computer graphics.

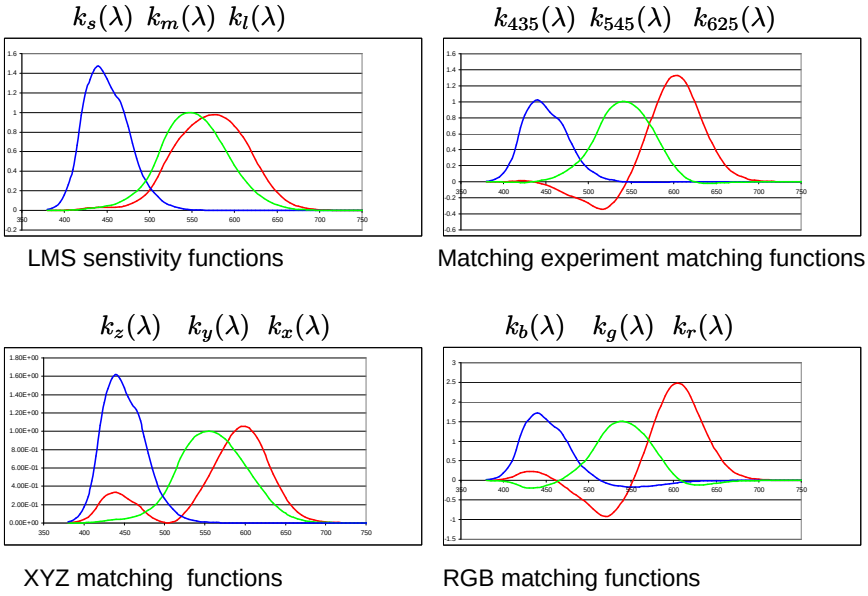


Figure 19.1: Sensitivity/matching functions.

## 19.1 Simple Bio-Physical Model

Visible light is electromagnetic radiation that falls roughly in the *wavelengths* $380 < \lambda < 770$, measured in nanometers. (You can just think of each wavelength as a different "physical flavor" of light). We will talk about two kinds of beams of lights. A *pure beam* $l_\lambda$ has one "unit" of light (measured in units of irradiance) of a specific wavelength $\lambda$. A *mixed beam* $l(\lambda)$ has different amounts of various wavelengths. These amounts are determined by the function $l(\cdot) : R \to R_+$, and are in units of spectral irradiance. The value is always non-negative since there is no "negative light".

The human eye has various kinds of light-sensitive cells on the retina. The *cone* cells give us the sensation of color. (Non color-blind) humans have three different kind of cones, which we will call short, medium and long (after the wavelengths of light they are most sensitive to). Associated with these three types of cones are three sensitivity functions $k_s(\lambda)$, $k_m(\lambda)$ and $k_l(\lambda)$. A response function describes how strongly one type of cone "responds" to pure beams of light of different wavelengths. For example, $k_s(\lambda)$ tells us how much a short-wavelength sensitive cone will respond to the pure beam $l_\lambda$. (See the upper left of Figure 19.1).

Since each pure beam of light results in three response values on the retina, one for each type of cone, we can visualize this response as a single point in a 3D space. Let us define a 3D linear space, with coordinates labeled $[S, M, L]^t$. Then for a fixed $\lambda$, we can draw the retinal response as a single vector with coordinates $[k_s(\lambda), k_m(\lambda), k_l(\lambda)]^t$.



Foundations of 3D Computer Graphics
S.J. Gortler

MIT Press, 2012

As we let $\lambda$ vary, such vectors will trace out a *lasso* curve in space (see the top row of Figure 19.7). The lasso curve is parametrized by $\lambda$. The lasso curve lies completely in the positive octant since all responses are positive. The curve both starts and ends at the origin since these extreme wavelengths are at the boundaries of the visible region, beyond which the responses are zero. The curve spends a short time on the $L$ axis (shown with blue tinted points) and finally comes close to the $S$ axis (shown in red). The curve never comes close to the $M$ axis, as there is no light that stimulates these cones alone.

In this simple model, we think of the $[S, M, L]^t$ coordinates of the light beam as describing the (retinal) color sensation produced by the light beam. We use the symbol $\vec{c}$ to represent a color itself, which we equate, for now, with an actual retinal event. Soon we will define color more rigorously. Thus, in Figure 19.7, we can think of each 3D vector as *potentially* representing some color. Vectors on the lasso curve are the *actual* colors of pure beams.

Within some ranges of intensity, the cones respond linearly to the light shining on them. Thus, for a mixed beam of light $l(\lambda)$, the three responses $[S, M, L]^t$ are

$$S = \int_{\Omega} d\lambda \;\; l(\lambda)\; k_s(\lambda) \tag{19.1}$$

$$M = \int_{\Omega} d\lambda \;\; l(\lambda)\; k_m(\lambda) \tag{19.2}$$

$$L = \int_{\Omega} d\lambda \;\; l(\lambda)\; k_l(\lambda) \tag{19.3}$$

where $\Omega = [380..770]$.

As we look at all possible mixed beams $l(\lambda)$, the resulting $[S, M, L]^t$ coordinates sweep out some set of vectors in 3D space. Since $l$ can be any positive function, the swept set is comprised of all positive linear combinations of vectors on the lasso curve. Thus, the swept set is the *convex cone* over the lasso curve, which we call the *color cone*. Vectors inside the cone represent actual achievable color sensations. Vectors outside the cone, such as the vertical axis do not arise as the sensation from any actual light beam, whether pure or composite.

To help visualize the cone, we have broken down its drawing in Figure 19.7 into a series of steps. In the second row, we have normalized the lasso curve, scaling each vector so $S + M + L = K$, for some constant $K$. Such a scaled lasso curve is called a *horseshoe curve*. We also add tails joining this horseshoe curve to the origin. In the third row, we add lines from the origin to the horseshoe curve. This is to try to give you a better feel for the shape of the color cone. Finally, in the fourth row, we place an opaque plane showing one slice of the color cone. On this plane, we also draw the actual colors that are on this slice and that are producible by linear combinations of R, G and B: red, green, and blue monitor elements. (This RGB space will be discussed in detail below). To draw the brightest colors, subject to these constraints, we have chosen the value of $K$ in $S + M + L = K$ such that the slice includes the color with RGB coordinates $[1, 0, 0]^t$. In wireframe, we show the RGB-cube, the set of colors that

can be achieved by combinations of red green and blue with coefficients in $[0..1]$.

There are an infinite number of vectors making up the lasso curve, certainly more than three! Thus, for vectors strictly inside the color cone, there are many ways to generate some fixed $[S, M, L]^t$ coordinates using positive linear combinations of vectors on the lasso curve. Each of these is equivalent to some light beam that produces this fixed response. Thus, there must be many physically distinct beams of light, with different amounts of each wavelengths, that generate the same color sensation. We call any two such beams *metamers*.

Here, we summarize the distinct data types we have just seen, as well some that we will soon see below

- A pure beam of light is denoted $l_\lambda$. A mixed beam of light is denoted $l(\lambda)$.
- A sensitivity function is denoted as $k(\lambda)$. We will later also call these *matching functions*.
- A retinal sensation of color is denoted by $\vec{c}$. Below, we will use three such colors to make a basis for color space.
- A color is represented by three coordinates, such as $[S, M, L]^t$. The coordinates of the observed color of a beam are calculated using the matching functions as in Equation (19.1).
- Below, we will also see a reflection function $r(\lambda)$ that describes the fraction of each wavelength that is reflected by some physical material.

### 19.1.1 Map of Color Space

At this point, we already have enough information to roughly map out the color cone.

Scales of vectors in the cone correspond to brightness changes in our perceived color sensation, and thus are not very interesting. (Though, when we dim an orange light, we actually perceive the color brown.) Thus it is convenient to normalize our color diagram by scale, obtaining a 2D drawing (see Figure 19.2). In this diagram, we have started with the slice of the cone from Figure 19.7. All displayed colors have then been scaled so that one of R, G, or B is at full strength. We cannot display colors outside of the drawn triangle using positive combinations of the R G and B display elements of a monitor. We say that such colors lay outside of the *gamut* of our display.

Colors along the boundary of the cone are vivid and are perceived as "saturated". Starting from the $L$ axis, and moving along the curved portion, we move along the rainbow colors from red to green to violet. These colors can only be achieved by pure beams. Additionally the color cone's boundary has a planar wedge (a line segment in the 2D figure). The colors on this wedge are the pinks and purples. They do not appear in the rainbow and can only be achieved by appropriately combining beams of red and violet. As as we circle around the boundary, we move through the different "hues" of color.



Foundations of 3D Computer Graphics
S.J. Gortler

MIT Press, 2012

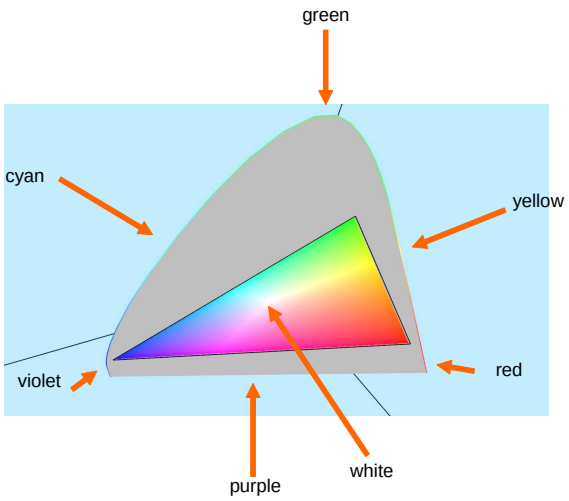


Figure 19.2: Two dimensional color diagram. Colors outside of the triangle are beyond the gamut of a computer display.

As we move in from the boundary towards the central region of the cone, the colors, while maintaining their hue, desaturate, becoming pastel and eventually grayish or whitish. (Though in our treatment we will not need to have a specific color formally selected as white).

This general description can be numerically formulated in the so called hue saturation value system of describing color.

## 19.2 Mathematical Model

The model just described in Section 19.1 was actually deduced in the 19th century using just a few perceptual experiments. They had no access to the technologies needed to study cells in an eye. This was an amazing feat. Here, we follow this original line of reasoning and explain how our color model can be deduced from the ground up with just the right kind of perceptual experiments. This will give us a more careful understanding of how to define color, and it will let us treat color space with all the tools of linear algebra, without reference to neural responses of any kind.

We start only with the basic knowledge from physics, that light beams can be described as wavelength distributions $l(\lambda)$, and the rough observation that distinct light distributions can sometimes appear indistinguishable to a human observer. In order to carefully study such metamerism, and specifically to avoid any effects that may occur when a human observes a complicated scene, we design an experimental setup such as



Foundations of 3D Computer Graphics
S.J. Gortler

MIT Press, 2012

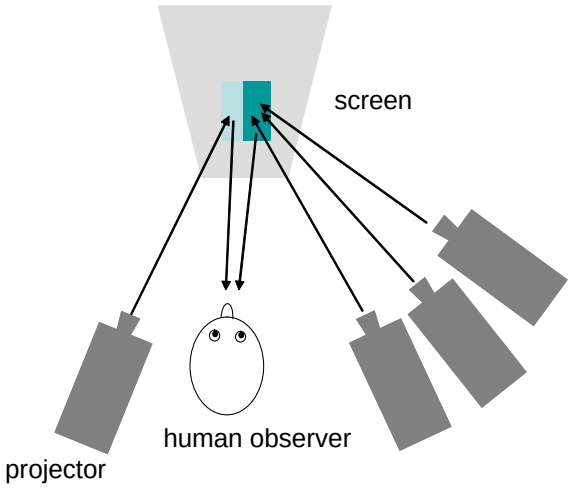


Figure 19.3: Basic color-matching experimental setup. Light projectors focus light beams with various wavelength distributions onto a large monochromatic screen in a way that forms two colored patches, each with a controlled wavelength distribution. Multiple projectors are placed on the right side so we can also test what happens when various light distributions are added together. A human observer is asked if the two patches can be distinguished by color, or not.

that shown in Figure 19.3. This allows us to present to an observer two light beams with known wavelength distributions. We can then ask the observer if these beams appear identical or different.

In our very first experiment, we test that the metameric relation is transitive (here we ignore the issue of just noticeable differences, and thresholding effects). In particular we find that, if $l_1(\lambda)$ is indistinguishable to $l_1'(\lambda)$, and $l_1'(\lambda)$ is indistinguishable to $l_1''(\lambda)$, then $l_1'(\lambda)$ will always be indistinguishable to $l_1''(\lambda)$.

Due to this transitivity, we actually *define* $\vec{c}(l_1(\lambda))$, "the color of the beam $l_1(\lambda)$", as the collection of light beams that are indistinguishable to a human observer from $l_1(\lambda)$. So in our case, we would have $\vec{c}(l_1(\lambda)) = \vec{c}(l_1'(\lambda)) = \vec{c}(l_1''(\lambda))$. Thus in our mathematical model, **a (retinal) color is an equivalence class of light beams**.

Ultimately, we would like to be able to treat the space of colors as a linear vector space. This, for example, would allow us to easily represent colors using coordinate vectors, and it would tell us how we could produce desired colors by mixing together various "primary" colors.

Our next step, then, is to figure out how to add two colors together. We know from physics that when two light beams, $l_1(\lambda)$ and $l_2(\lambda)$, are added together, they simply form a combined beam with light distribution $l_1(\lambda)+l_2(\lambda)$. Thus, we attempt to define



Foundations of 3D Computer Graphics
S.J. Gortler

MIT Press, 2012

the *addition* of two colors, as the color of the addition of two beams.

$$\vec{c}(l_1(\lambda)) + \vec{c}(l_2(\lambda)) := \vec{c}(l_1(\lambda) + l_2(\lambda))$$

For this to be well defined, we must experimentally verify that it does not make a difference which beam we choose as representative for each color. In particular, if $\vec{c}(l_1(\lambda)) = \vec{c}(l_1'(\lambda))$, then we must verify (again using our setup of Figure 19.3) that, for all $l_2(\lambda)$, we have $\vec{c}(l_1(\lambda) + l_2(\lambda)) = \vec{c}(l_1'(\lambda) + l_2(\lambda))$, i.e., we must test that the beam $l_1(\lambda) + l_2(\lambda)$ is indistinguishable to $l_1'(\lambda) + l_2(\lambda)$. This property is indeed confirmed by experiment.

Our next step is to try to define what it means to multiply a color by a non-negative real number $\alpha$. Again, since we can multiply a light beam by a positive scalar, we try the definition

$$\alpha\vec{c}(l_1(\lambda)) := \vec{c}(\alpha l_1(\lambda)) \tag{19.4}$$

Again, we need to verify that the behavior of this operation does not depend on our choice of beam. Thus when $\vec{c}(l_1(\lambda)) = \vec{c}(l_1'(\lambda))$ we must verify that for all $\alpha$ we have $\vec{c}(\alpha l_1(\lambda)) = \vec{c}(\alpha l_1'(\lambda))$, i.e., we must test that the beam $\alpha l_1(\lambda)$ is indistinguishable to $\alpha l_1'(\lambda)$. This property is also confirmed by experiment.

### 19.2.1 One Technical Detail

In a real vector space, we are able to multiply a vector by a negative real number. If we try this on our color representation, when we get $-\vec{c}(l_1) := \vec{c}(-l_1)$. This is undefined since there is no negative light.

Still, it would be nice to be able to treat colors as vectors and to apply the full power of linear algebra to them. In particular, as we will see below, due to the shape of the color horseshoe, we will not be able to represent all colors as positive combinations of just three colors. To do that, we will really need negative combinations as well.

Our mathematical solution is first define a suitable notion of subtraction. The basic idea behind this subtraction operation can be summarized as follows: when we say $\vec{c}_1 - \vec{c}_2 = \vec{c}_3$, we really mean $\vec{c}_1 = \vec{c}_3 + \vec{c}_2$. In other words, subtraction from one side of an equation is just the same as adding that term to the other side. This addition is something real, that we already understand! With this notion of subtraction, we can give meaning to "negative" colors. By adding together actual and negative colors, we can get a full linear space, which we can call *extended color space*.

More formally, let us call any of our original equivalence classes of light beams using the term: *actual color*. Let us define an *extended color* as a formal expression of the form

$$\vec{c}_1 - \vec{c}_2$$

where the $\vec{c}$ are actual colors. We define two extended colors $\vec{c}_1 - \vec{c}_2$ and $\vec{c}_3 - \vec{c}_4$, to be equivalent if $\vec{c}_1 + \vec{c}_4 = \vec{c}_3 + \vec{c}_2$, where the last expression is an equation about actual



Foundations of 3D Computer Graphics
S.J. Gortler

MIT Press, 2012

colors, and thus well defined. Any extended color that is not an actual color will be called an *imaginary color*.

We can now define all of our vector operations in the obvious way. Multiplication by $-1$ is $-(\vec{c}_1 - \vec{c}_2) := (\vec{c}_2 - \vec{c}_1)$ and addition is $(\vec{c}_1 - \vec{c}_2) + (\vec{c}_3 - \vec{c}_4) := (\vec{c}_1 + \vec{c}_3) - (\vec{c}_2 + \vec{c}_4)$. With these operations, we indeed have a linear space of extended colors!

Finally, to keep actual distinguishable colors from collapsing to the same extended color during this construction, we need to verify that our actual colors satisfy the *cancellation property*. This property states that if $\vec{c}(l_1(\lambda)) + \vec{c}(l_2(\lambda)) = \vec{c}(l_1'(\lambda)) + \vec{c}(l_2(\lambda))$ then $\vec{c}(l_1(\lambda)) = \vec{c}(l_1'(\lambda))$. Again we verify this by experiment.

As a result, we now have a real vector space of extended colors, as well as an embedding of the actual colors within this space. From now on, we will use the symbol $\vec{c}$ to refer to any extended color, and will typically drop the term "extended". Additionally, we can interpret $\vec{c}(l(\lambda))$ as a linear map from the space of light distributions to color space.

We do not yet know dimension of color space, (but we will soon establish that it is three). We can now go back to Figure 19.7 and think of it as a picture of extended color space. Vectors inside the cone are actual colors, while vectors outside the cone are imaginary colors. The vector, for example, represented with coordinates $[0, 1, 0]^t$ is an imaginary color.

## 19.3 Color Matching

The goal of the *color matching experiment* is to establish that the dimension of the space of colors is three. Additionally, it will give us (similar to Equation (19.1)) a computational form for mapping a light beam $l(\lambda)$ to its color coordinates in a specific basis.

Using the setup of Figure 19.3, the observer watches two screens. On the left side of the screen they are shown a pure *test beam* $l_\lambda$ of some fixed wavelength $\lambda$. On the right side of the screen they observe a light that is made up of positive combinations of three pure *matching beams*, with wavelengths $435$, $545$ and $625$ nanometers. The observer's goal is to adjust three knobs on the right side, controlling the intensities of the matching beams, so that the weighted combination of the three matching beams is indistinguishable to the test beam. For a fixed $\lambda$, and referring to the knob settings as $k_{435}(\lambda)$, $k_{545}(\lambda)$ and $k_{625}(\lambda)$, the goal is to set these knobs such that the beam $k_{435}(\lambda)l_{435} + k_{545}(\lambda)l_{545} + k_{625}(\lambda)l_{625}$ is a metamer with $l_\lambda$. If the user cannot succeed, then they are allowed to move one or more of the matching beams over to the left side and combine them with the test beam instead. In the mathematics of extended color space, this is the same as allowing some of the the scalar values $k(\lambda)$ to go negative.

This process is repeated for all $\lambda$ in the visual range. When the matching experiment is performed, we discover that the user can indeed succeed in obtaining a match



Foundations of 3D Computer Graphics
S.J. Gortler

MIT Press, 2012

for *all* visible wavelengths.

Moreover, the experiment gives us the three so-called matching functions $k_{435}(\lambda)$, $k_{545}(\lambda)$ and $k_{625}(\lambda)$, shown in the upper right of Figure 19.1. Notice that, at each of the wavelengths $435$, $545$, and $625$, one of the matching functions is set to $1$, while the other two are set to $0$.

We can summarize the result of the experiment as

$$\vec{c}(l_\lambda) = [\vec{c}(l_{435})\ \vec{c}(l_{545})\ \vec{c}(l_{645})] \begin{bmatrix} k_{435}(\lambda) \\ k_{545}(\lambda) \\ k_{625}(\lambda) \end{bmatrix}$$

Using some reasonable continuity assumptions about the linear map $\vec{c}$, we can upgrade this equation to apply to all mixed beans as well. Doing so, we obtain

$$\vec{c}(l(\lambda)) = [\vec{c}(l_{435})\ \vec{c}(l_{545})\ \vec{c}(l_{645})] \begin{bmatrix} \int_\Omega d\lambda\ \ l(\lambda)\ k_{435}(\lambda) \\ \int_\Omega d\lambda\ \ l(\lambda)\ k_{545}(\lambda) \\ \int_\Omega d\lambda\ \ l(\lambda)\ k_{625}(\lambda) \end{bmatrix} \tag{19.5}$$

Informally, this equation corresponds to the idea that each mixed beam is really just an (uncountable) linear combination of pure beams.

From we can conclude

- Color space is three dimensional.
- $[\vec{c}(l_{435})\ \vec{c}(l_{545})\ \vec{c}(l_{645})]$ forms a basis for this space.
- The matching functions can be used to give us the coordinates of any light distribution with respect to this basis.

As we did with the LMS color space, We can visualize this color space in Figure 19.8. Notice that, in this case, the lasso curve passes through each of the axes in turn, as our basis colors are mono-chromatic. Note though that, in this basis, the lasso curve does leave the first octant.

## 19.4 Bases

As any vector space, color space can be described using many different bases. Starting with Equation (19.5) we can insert any (non singular) 3-by-3 matrix $M$ and its inverse to obtain

$$\begin{aligned} \vec{c}(l(\lambda)) &= \left([\vec{c}(l_{435})\ \vec{c}(l_{545})\ \vec{c}(l_{645})]M^{-1}\right) \left( M \begin{bmatrix} \int_\Omega d\lambda\ \ l(\lambda)\ k_{435}(\lambda) \\ \int_\Omega d\lambda\ \ l(\lambda)\ k_{545}(\lambda) \\ \int_\Omega d\lambda\ \ l(\lambda)\ k_{625}(\lambda) \end{bmatrix} \right) \\ &= [\vec{c}_1\ \vec{c}_2\ \vec{c}_3] \begin{bmatrix} \int_\Omega d\lambda\ \ l(\lambda)\ k_1(\lambda) \\ \int_\Omega d\lambda\ \ l(\lambda)\ k_2(\lambda) \\ \int_\Omega d\lambda\ \ l(\lambda)\ k_3(\lambda) \end{bmatrix} \end{aligned} \tag{19.6}$$



Foundations of 3D Computer Graphics
S.J. Gortler

MIT Press, 2012

where the $\vec{c}_i$ describe a new color basis defined as

$$[\vec{c}_1\ \vec{c}_2\ \vec{c}_3] = [\vec{c}(l_{435})\ \vec{c}(l_{545})\ \vec{c}(l_{645})]M^{-1}$$

and the $k(\lambda)$ functions form the new associated matching functions, defined by

$$\begin{bmatrix} k_1(\lambda) \\ k_2(\lambda) \\ k_3(\lambda) \end{bmatrix} = M \begin{bmatrix} k_{435}(\lambda) \\ k_{545}(\lambda) \\ k_{625}(\lambda) \end{bmatrix} \tag{19.7}$$

Thus, there are three main conceptual ways to specify a basis for color space:

- Starting from any fixed basis for color space, such as $[\vec{c}(l_{435})\ \vec{c}(l_{545})\ \vec{c}(l_{645})]$, we can describe a new basis relative to the fixed basis by specifying an invertible 3-by-3 matrix $M$.
- We can directly specify three (non-coplanar) actual colors $\vec{c}_i$. Each such $\vec{c}_i$ can be specified by some light beam $l_i(\lambda)$ that generates it. (We can then plug each such $l_i(\lambda)$ into the right hand side of Equation (19.5) to obtain its coordinates with respect to $[\vec{c}(l_{435})\ \vec{c}(l_{545})\ \vec{c}(l_{645})]$. This fully determines the change of basis matrix $M$.)
- We can directly specify three new matching functions. To be valid matching functions, they must arise from a basis change like Equation (19.6), and so each matching function must be some linear combination of $k_{435}(\lambda)$, $k_{545}(\lambda)$ and $k_{625}(\lambda)$ as in Equation (19.7). If we attempt to use matching functions that are not of this form, they will not preserve metamerism; light beams that are indistinguishable to a human may map to different coordinate vectors, and vice versa. Ideally, the color sensors for a digital camera should be of this form, so that the camera can truly capture color, i.e., respect metamerism. Additionally, the sensitivity functions of a manufactured camera must also be everywhere non-negative.

Besides $[\vec{c}(l_{435})\ \vec{c}(l_{545})\ \vec{c}(l_{645})]$, we have already seen another basis for color space. In particular, the matching functions of Equation (19.1) describe a basis for color space where the coordinates of a color are called $[S, M, L]^t$. The actual basis is made up of three colors we can call $[\vec{c}_s, \vec{c}_m, \vec{c}_l]$. The color $\vec{c}_m$ is in fact an imaginary color, as there is no real light beam with LMS color coordinates $[0, 1, 0]^t$.

### 19.4.1 Gamut

Suppose we want a basis where all actual colors have non-negative coordinates, and thus, where the lasso curve never leaves the first octant. Then we find that **at least one of the basis vectors defining this octant must lie outside of the cone of actual colors**. Such a basis vector must be an imaginary color. This is due simply to the shape



Foundations of 3D Computer Graphics
S.J. Gortler

MIT Press, 2012

of the lasso curve itself; we cannot find three vectors that both hit the lasso curve and contain the entire curve in their positive span.

Conversely, if all of our basis vectors are actual colors, and thus within the color cone, then there must be some actual colors that cannot be written with non-negative coordinates in this basis. We say that such colors lie outside the *gamut* of this color space.

### 19.4.2 Specific Bases

The central standard basis used for color space is the called the XYZ basis. It is specified by the three matching functions called $k_x(\lambda)$, $k_y(\lambda)$ and $k_z(\lambda)$, shown in the bottom left of Figure 19.1. The coordinates for some color with respect to this basis is given by a coordinate vector that we call $[X, Y, Z]^t$. This 3D color basis is shown in Figure 19.9. The bottom row shows the $X + Y + Z = K$ plane of the color cone. This is the typical 2D figure used to visualize color space.

These particular matching functions were chosen such that they are always positive, and so that the Y-coordinate of a color represents its overall perceived "luminance". Thus, $Y$ is often used as a black and white representation of the color. The associated basis $[\vec{c}_x, \vec{c}_y, \vec{c}_z]$ is made up of three imaginary colors; the axes in Figure 19.9 are outside of the color cone.

Throughout this book, we have been using RGB coordinates to describe colors. In fact, there are a variety of different color spaces that use this name. The specific RGB color space currently in use is the *Rec. 709 RGB space*. (see Figure 19.10).

In this case the basis $[\vec{c}_r, \vec{c}_g, \vec{c}_b]$ is made up of three actual colors intended to match the colors of the three phosphors of an ideal monitor/tv display. Colors with non-negative RGB coordinates can be produced on a monitor and are said to lie inside the *gamut* of the color space. These colors are in the first octant of the Figure. But similar to the case of $[\vec{c}(l_{435})\ \ \vec{c}(l_{545})\ \ \vec{c}(l_{645})]$, there exist actual colors with some negative RGB coordinates. **Such colors cannot be produced on a monitor.** Additionally, on a monitor, each phosphor maxes out at "1", which also limits the achievable outputs.

An image that has colors outside of the gamut must somehow be mapped into the gamut for display. The simplest solution for this is to simply clamp all negative values at $0$. There are also more sophisticated methods for gamut mapping that will be beyond our scope.

In Section 19.7.2, we will describe another commonly encountered color space called sRGB. As we will see, this is not a linear color space.

## 19.5 Reflection Modeling

When a beam of light $i(\lambda)$ from an illumination source hits a surface, some of that light is absorbed and some reflected. The fraction of reflected light depends on the physical properties of the surface's material. Let us specify how much of each wavelength is reflected using a reflectance function $r(\lambda)$. In this case, we can model the light beam reflecting off the surface using per-wavelength multiplication

$$l(\lambda) = i(\lambda)r(\lambda)$$

(Note: this does not model all types of interactions between a light and a surface, for example florescence. Additionally, in this discussion, we are not concerning ourselves with the dependence of $r(\lambda)$ on the angles of entering or exiting light, as will be done in Chapter 21.) This multiplication happens on a per-wavelength basis, and cannot be simulated exactly in a 3D color space. Indeed, two materials may reflect metameric beams under one illuminant, but may produce distinguishable beams under a second illuminant:

$$\vec{c}(i_1(\lambda)r_a(\lambda)) = \vec{c}(i_1(\lambda)r_b(\lambda)) \quad \not\Leftrightarrow \quad \vec{c}(i_2(\lambda)r_a(\lambda)) = \vec{c}(i_2(\lambda)r_b(\lambda))$$

As such, in some rendering situations, it is important to model this spectral dependence in reflection. More typically, we ignore this issue, and model the illuminant by three, say RGB, color coordinates (throwing away the spectral information about $i(\lambda)$), and likewise use three reflectance "coefficients" to model the surface's reflectance property.

### 19.5.1 White Balance

Given a fixed scene, if we alter the illuminants, then the colors in an image will change as well. For example, if we switch from a fluorescent to an incandescent bulb, the colors observed by a camera will all move towards yellow. Often, we wish to adjust the image colors in order to approximate the image that would have been taken under a chosen "canonical illuminant" (say daylight). This process is called white balancing. It is not a basis change, but an actual transformation performed on all of the colors. The simplest such kind of transform allows the user to independently scale the R,G and B coordinates with three gain factors.

As just described, we cannot hope to always succeed in producing the true picture of the scene under the canonical illuminant, since we have already lost the spectral information when creating the initial image. Indeed, some objects that should appear different under the canonical illuminant may be metameric under the current illuminant, and have the exact same color coordinates in the current image. No amount of simple white balancing can undo this.



Foundations of 3D Computer Graphics
S.J. Gortler

MIT Press, 2012

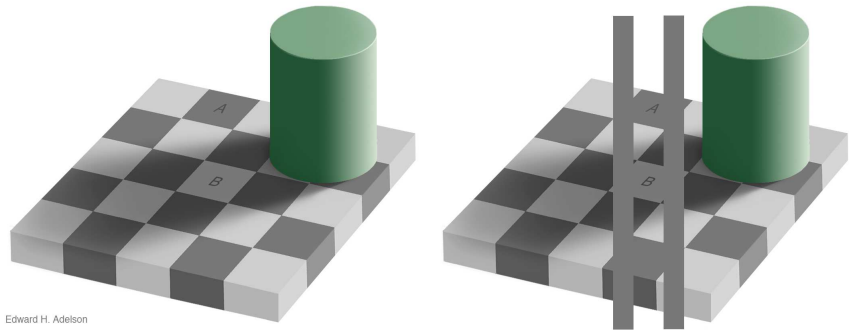


Figure 19.4: The squares marked A and B are in fact the same retinal shade of gray, but are perceived differently due to our visual processing and local adaptation. From [1], ©Edward Adelson.

## 19.6 Adaptation

The color data from the retina undergoes significant processing in the visual system, and humans by no means directly perceive the raw retinal color coordinates of an observed light beam. This processing results in a great deal of normalization; adapting to global and local trends across the field of view.

When the illuminant changes, say from sunlight to overcast sky, each of the directly observed color coordinates on the retina may undergo drastic changes. But these drastic changes are not ultimately perceived, and the colors for each object remain significantly "constant". For example, a scary tiger will be perceived as yellow under a surprisingly wide array of illuminants (impelling us to run). This phenomenon is called *color constancy*. As per our discussion of white balance, no such color constancy mechanism can be expected to be perfect, as too much spectral information has been thrown away by the process that converts incoming spectral beams into triplets of cone responses in the retina. But this process works to a great degree, which allows us to think about a material (tiger's fur) as actually possessing a color (scary orange).

Even when only a local region of the field of view undergoes an illumination change (say some part goes into shadow) our visual processing may adapt differently in this region, again keeping the end perceived colors closely tied with the actual materials observed. (See for example Figure 19.4). This process is not yet fully understood.

When we take a picture under some illuminant, but later view the picture under a different ambient illuminant, the viewer's adaptation state is affected by both the light coming from the image, as well as the light from the surrounding room. Due to the effect of the room's light, the colors in the picture can ultimately "look wrong". This is, in part, why we need to do the white balancing described above.



Foundations of 3D Computer Graphics
S.J. Gortler

MIT Press, 2012

## 19.7 Non Linear Color

We have seen that retinal color can be modeled as a three dimensional linear space. In this section, we will see that there are also reasons to use a different set of retinal color representations that are not related linearly to our previous color coordinates.

### 19.7.1 Perceptual Distance

The Euclidean distance between two colors in any linear color space is not a good predictor as to how "different" they will appear to a human observer. For example, humans are much more sensitive to changes in dark colors than they are to bright ones. Various color representations have been designed that offer a better match to perceived color distances in humans. The mappings from a linear color space to such a color representation is non-linear. Even so, we will still refer to such representations as "color coordinates".

For example, one such set of coordinates is called $L^*ab$ coordinates. The $L^*$ coordinate is called "lightness" and is computed (except for very small values) as

$$L^* = 116\left(\frac{Y}{Y_n}\right)^{\frac{1}{3}} - 16 \tag{19.8}$$

where $Y$ is the second coordinate in the XYZ basis, and $Y_n$ is some normalizing factor. We will not go into the computation of the $a$ and $b$ coordinates in this representation.

There are many uses for such a space. In particular, if we are using a fixed point representation with 8 or fewer bits per coordinate, we are better off storing our data in a perceptually uniform space. When a continuum of $Y$ values is bucketed into 256 evenly spaced bins, there will be significant visual gaps between dark colors. In $L^*$ coordinates, tighter bins are used in the dark region, solving this problem. There will be correspondingly fewer bins for the brighter colors, but these gaps are not perceivable.

### 19.7.2 Gamma Correction

Gamma correction involves a transformation that looks similar to the power operator of Equation (19.8). It was used originally to account for non-linearities in CRT devices, but remains in use, in part due to its better usage of fixed point representations.

**Origins of Gamma**: In days of yore, computer imagery was displayed on cathode ray tubes (CRTs). Each pixel on such a display was driven by three voltages, say $(R', G', B')$. Letting the outgoing light from this pixel have a color with coordinates $[R, G, B]^t$, these outgoing coordinates were roughly

$$\begin{aligned} R &= (R')^{\frac{1}{.45}} \\ G &= (G')^{\frac{1}{.45}} \\ B &= (B')^{\frac{1}{.45}} \end{aligned}$$



Foundations of 3D Computer Graphics
S.J. Gortler

MIT Press, 2012

Figure 19.5: The data in the top image is a linear ramp of colors, thus displaying (on a monitor) equally spaced bins in $[R', G', B']^t$ coordinates. In the lower image, a linear ramp has been gamma corrected before being stored. Thus, displaying (on a monitor) equally spaced bins in $[R, G, B]^t$ coordinates. This should appear to move quickly out of the dark regime, and spend more buckets on bright values.

Thus, if we wanted to obtain some specific $[R, G, B]^t$ output from a pixel, we needed to drive it with voltages:

$$R' = R^{.45} \tag{19.9}$$
$$G' = G^{.45} \tag{19.10}$$
$$B' = B^{.45} \tag{19.11}$$

Such $[R', G', B']^t$ values are called the *Gamma Corrected* RGB coordinates of a color. The (') notates that these are nonlinear color coordinates.

**Current use of Gamma**: Similar to $L^*ab$ color coordinates, gamma corrected colors have better perceptual uniformity than linear color coordinates, and thus are very useful for digital color representation (see Figure 19.5). In particular, popular image compression techniques, such as JPEG, start with colors represented in $[R', G', B']^t$, and then apply a linear transform to obtain yet a new kind of coordinates called $[Y', C'_B, C'_R]^t$. (Note that this $Y'$ is not related to $Y$ through a simple power equation).

A related but slightly more involved non-linear transform can be applied to $[R, G, B]^t$, instead of Equation (19.9), to obtain sRGB coordinates, called $[R'_{\text{srgb}}, G'_{\text{srgb}}, B'_{\text{srgb}}]^t$. Modern LCD displays are programmed to assume input in these coordinates.

### 19.7.3 Quantization

The sRGB coordinates in the real range $[0..1]$ must be represented numerically. This is often done (say in a framebuffer or file format) in a fixed point representation with values $[0..255]$. In C, this is done using an `unsigned char`. We can specify the relationship between such quantized values and real color coordinates (for say the red coordinate) by

$$\texttt{byteR} = \texttt{round}(\texttt{realR} * 255);$$



Foundations of 3D Computer Graphics
S.J. Gortler

MIT Press, 2012


Figure 19.6: Two different mappings between real and byte values. Going from real to byte, we quantize each real range to the shown integer value. Going from byte to real we use the small cyan arrows.

$$\texttt{realR} = \texttt{byteR}/255.0;$$

Note that, for any setting of byteR, if we transform to the real representation and then back to the byte representation, we get back the value we started with. An alternative relationship satisfying this property can be imposed using the expressions:

$$\texttt{byteR} = \texttt{round}(\texttt{f} >= 1.0\ ?\ 255\ :\ (\texttt{realR} * 256) - .5);$$
$$\texttt{realR} = (\texttt{byteR} + .5)/\ 256.0;$$

In this representation, and unlike the one above, the real bins quantized to byte values are all the same size. But the byte values of $0$ and $255$ do not map respectively to $0$ and $1$. (See Figure 19.6).

### 19.7.4 Gamma and Graphics

On one hand, images are typically stored in gamma corrected coordinates and the monitor screen is expecting colors in gamma corrected coordinates. On the other hand, computer graphics simulates processes that are linearly related to light beams. As such, most computer graphics computations should be done in a linear color representation, such as our $[R, G, B]^t$ space. For example, we can approximately model reflectance in $[R, G, B]^t$. Other rendering steps, such as modeling transparency, as well as blending of color values for anti-aliasing, also model processes that are linear in light beams, and thus should be done with linear color coordinates. In digital photography, white balance should ideally be performed in a linear color space. This discrepancy has been at the root of much confusion and hackery over the years.

The situation has improved recently. In current versions of OpenGL we can request an sRGB frame buffer using the call `glEnable(GL_FRAMEBUFFER_SRGB)`. Then



Foundations of 3D Computer Graphics
S.J. Gortler

MIT Press, 2012

we can pass linear $[R, G, B]^t$ values out from the fragment shader, and they will be gamma corrected into the sRGB format before begin sent to the screen.

Additionally, for texture mapping, we can specify that the image being input to a texture is in sRGB format. This is done using the call `glTexImage2D(GL_TEXTURE_2D, 0, GL_SRGB, twidth, theight, 0, GL_RGB, GL_UNSIGNED_BYTE, pixdata)` Whenever this texture is accessed in a fragment shader, the data is first converted to linear $[R, G, B]^t$ coordinates, before given to the shader.

# Exercises

**Ex. 48 —** Given a computer screen with three kinds of color elements, can all (actual) colors be produced by the display?

**Ex. 49 —** Given a camera with three matching/sensitivity functions that are (linearly independent) linear combinations of the $k_x, k_y, k_z$ matching functions, can all actual colors be captured by this camera?

**Ex. 50 —** Suppose that the human $k_s$,$k_m$, and $k_l$ sensitivity functions were of a different form, such that there did in fact exist three light distributions with LMS color coordinates $[1, 0, 0]^t$, $[0, 1, 0]^t$ and $[0, 0, 1]^t$, respectively. What would this imply about the shape of the space of actual colors? Would this impact your answer to Exercise 48?

**Ex. 51 —** Suppose that we are given the following matrix equation to change from $[A, B, C]^t$ color coordinates to $[D, E, F]^t$ coordinates:

$$\begin{bmatrix} D \\ E \\ F \end{bmatrix} = N \begin{bmatrix} A \\ B \\ C \end{bmatrix}$$

Also, suppose we are given the following matrix equation relating the matching functions

$$\begin{bmatrix} k_h(\lambda) \\ k_i(\lambda) \\ k_j(\lambda) \end{bmatrix} = Q \begin{bmatrix} k_a(\lambda) \\ k_b(\lambda) \\ k_c(\lambda) \end{bmatrix}$$

What matrix equation can we write down to express the relation between $[D, E, F]^t$ coordinates and $[H, I, J]^t$ coordinates?

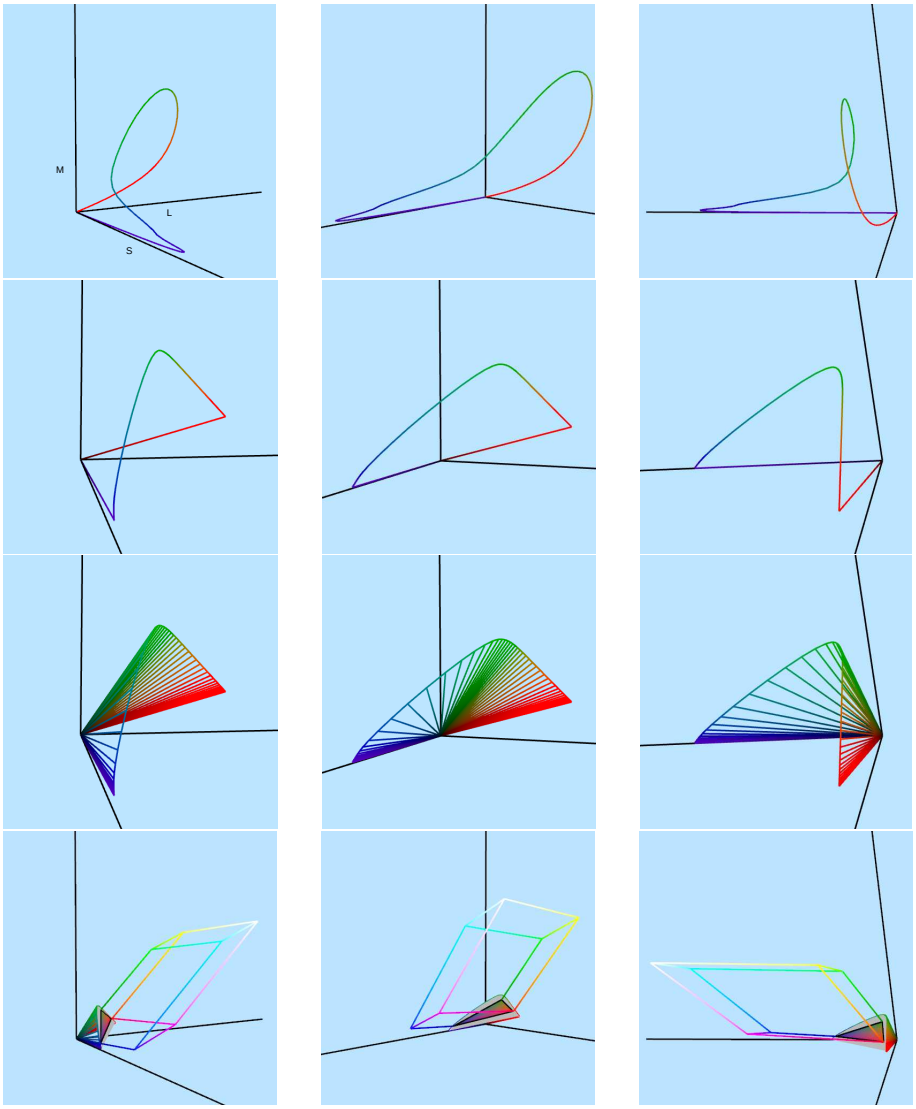


Figure 19.7: LMS color space: Each column shows a different view. First row: The lasso curve plotted in LMS coordinates. Second row: A "normalized" lasso curve is a horseshoe curve. Third row: rays connecting the horseshoe curve to the origin. Fourth row: A slice of the convex cone over the lasso. The triangle shows actual colors in this slice. They are representable as positive sums of monitor colors R, G, and B. The rest of the RGB color cube is shown in wireframe.



Foundations of 3D Computer Graphics
S.J. Gortler

MIT Press, 2012

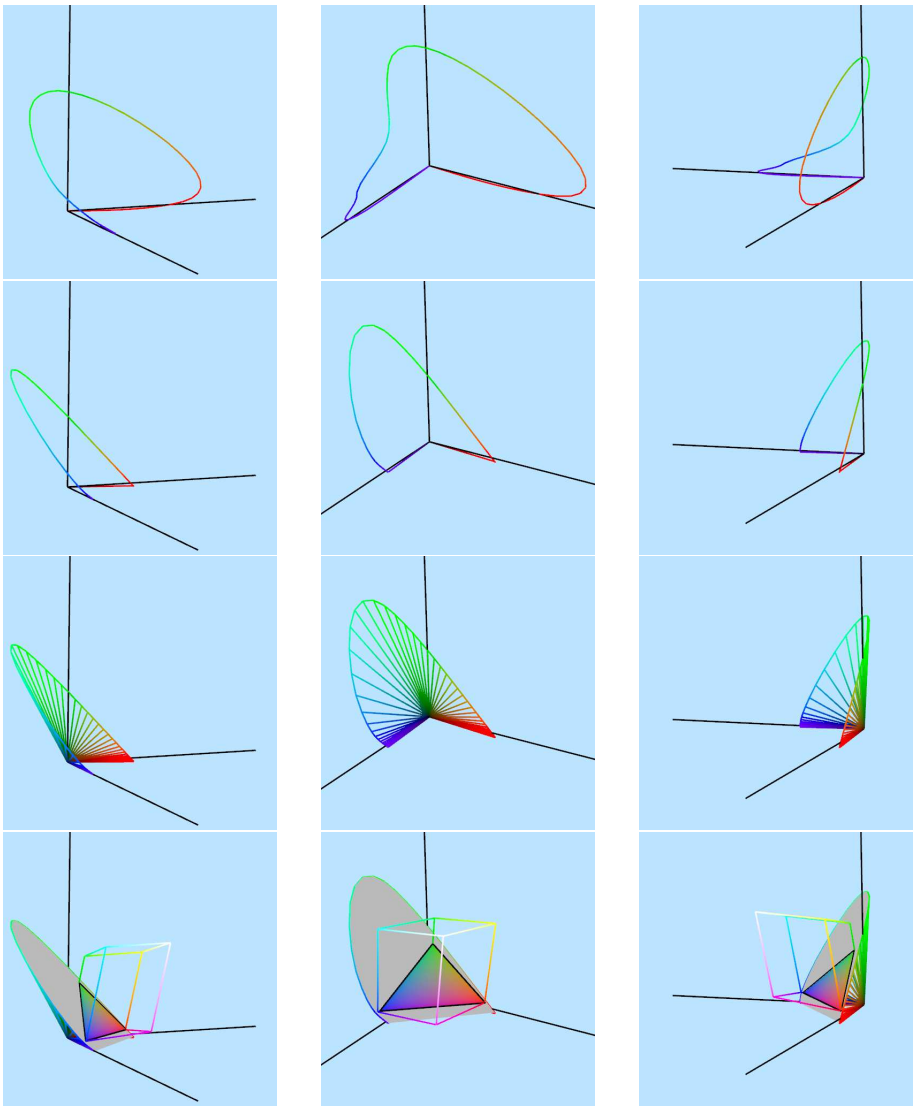

Figure 19.8: The color space arising from the matching experiment.



Foundations of 3D Computer Graphics
S.J. Gortler

MIT Press, 2012

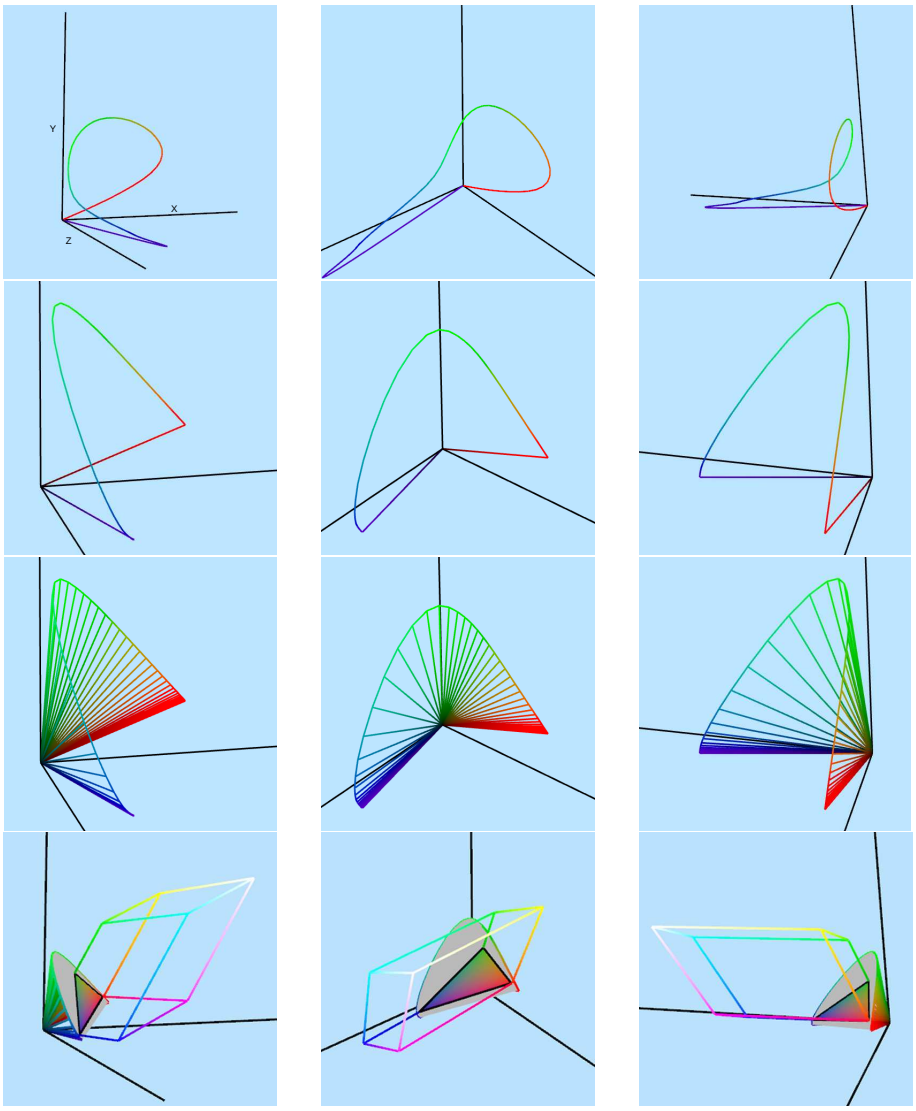


Figure 19.9: The XYZ color space.



Foundations of 3D Computer Graphics
S.J. Gortler

MIT Press, 2012

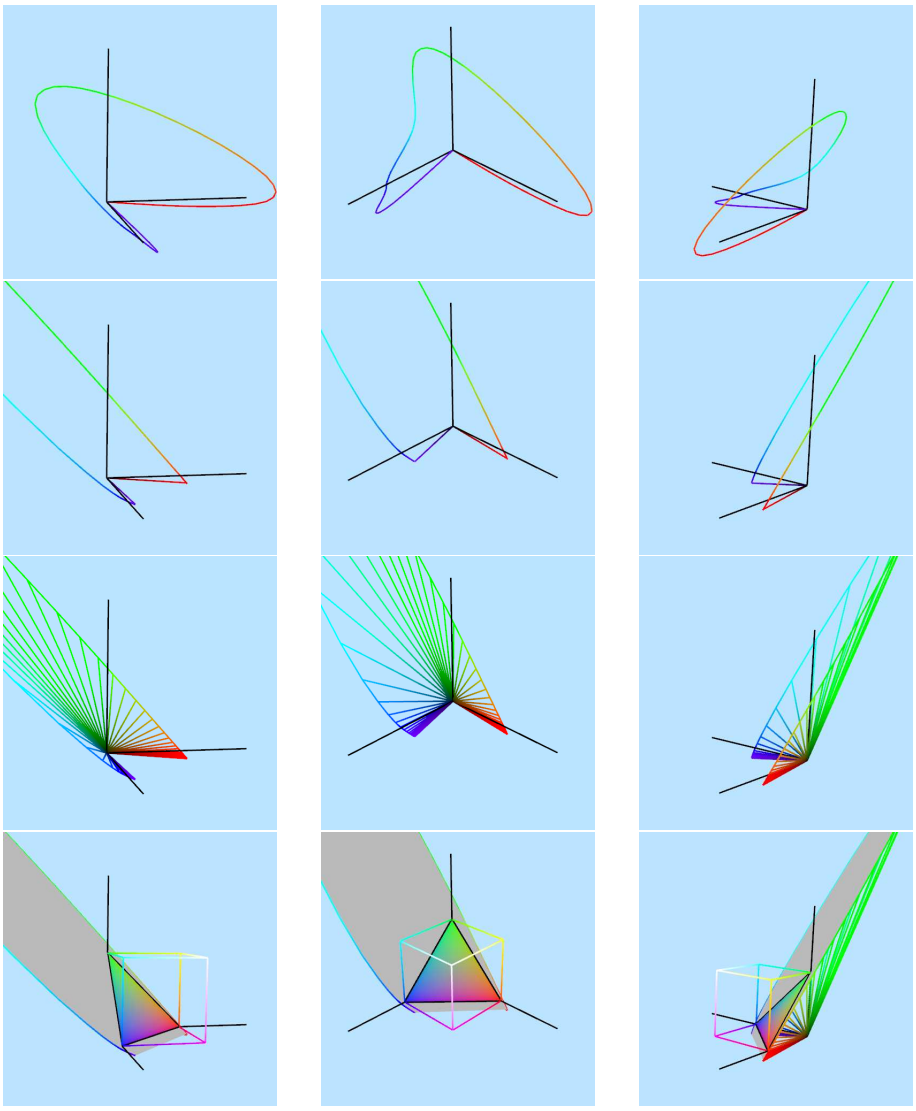

Figure 19.10: The RGB color space.