PADERBORN UNIVERSITY
The University for the Information Society

Department of Electrical Engineering, Computer Science, and Mathematics
Institute of Computer Science
Computer Graphics, Visualization, and Image Processing
Fürstenallee 11
33102 Paderborn

Conceptualization and Prototypical Implementation in WebGL of an Exercise in Color Theory

Thesis
in Candidacy for the Degree of
Bachelor of Science

by
Lukas Stratmann

First Examiner
Prof. Dr. Gitta Domik-Kienegger
Second Examiner
Prof. Dr. Holger Karl
Advisor
Sabrina Heppner

Paderborn, 2017-01-30

Declaration of Authorship

I hereby certify that this thesis has been composed by me and is based on my own work, unless stated otherwise. No other person's work has been used without due acknowledgement in this thesis. All references and verbatim extracts have been quoted, and all sources of information, including graphs and data sets, have been specifically acknowledged.

Place, date

Signature

Contents

List of Figures

List of Tables

Listings

1 Introduction

Understanding digital color is important for a wide variety of people and their occupations, including photographers, illustrators, CG artists, web developers, as well as people working in data visualization, or on human-machine interfaces. To understand the relationship between color models and to ideally be able to convert between them has advantages when one is subjected to numerical representations of colors. For example, when reading a CSS file, there would hardly be any need for typing each value into a color picker or to use a specialized integrated development environment (IDE) to get an impression of what each color might look like. Furthermore, a conceivable disadvantage for programmers of knowing only the RGB model, for example, is a tendency to pick exaggerated, bright and saturated colors.

1.1 Problem

Up to this point, students in Paderborn attending the lecture for computer graphics have had the opportunity to practice color models and learn about color theory using a specialized module of a project called SIMBA [SD03, pp. 407ff.]. The project was funded by the German Federal Ministry of Education and Research (Bundesministerium für Bildung und Forschung, BMBF) from 2001 to 2003. SIMBA is an acronym for „Schlüsselkonzepte der Informatik in verteilten multimedialen Bausteinen unter besonderer Berücksichtigung spezifischer Lerninteressen von Frauen“, which roughly translates to key concepts of computer science in distributed multimedia modules under special consideration of specific learning interests of women. It has been noted that such specific learning interests do not necessarily exist [Tig08, p. 166], yet the module for color theory received positive feedback from female and male students alike [Tig08, pp. 184ff.].

Unfortunately, the tool relies heavily on Java Applets. While Applets, up to only a few years ago, used to be a useful means to share small applications over the Internet without the need to explicitly download and install yet another program, there now exist more practical alternatives. Due to stricter security measures in web browsers, the current Java Applet for practicing color models has become quite difficult to run. It now requires adjustments in the plug-in settings and, in the case of Firefox 50.1, the dismissal of several warning messages.

1.2 Aim

The aim of this thesis is to implement a new studying tool for color theory. In turn, the aim for this tool is to convey a clear and precise understanding of color models and color spaces. It should do so in a way that is effective enough so that students will need little to no additional literature for grasping the essential information. The software itself should be usable and work reliably on desktop computers as well as on smartphones and tablets.

1.3 Approach

It was decided to move to an implementation in WebGL and therefore also in Javascript (JS). Approaching the problem like this removes the need for the user to have an implementation of Java installed. Neither do they require any other additional software, such as browser plug-ins, that does not come pre-installed with most desktop operating systems. A JS/WebGL implementation also opens up the possibility to run the same application on mobile devices like smartphones and tablets.

As the product to be developed is a piece of educational software, the information that will be presented should be well-founded. These foundations are researched in Chapter 2. Based on this information, Chapter 3 explores a concept for the product and its components. In Chapter 4, relevant details of the implementation of this concept are discussed. Lastly, Chapter 5 presents a method and the results of an evaluation of the prototype.

2 Fundamentals

In this chapter, the fundamentals of color theory with relevance to the aim of this thesis will be explored. This includes the color models featured in the exercises of the SIMBA project, namely RGB, HSL, and HSV. Furthermore, the CIE 1931 XYZ color space is studied as the basis for objective definitions of other color spaces. Munsell and CIELAB are listed as an extension to the notion of HSL and HSV of providing a color model based on the perception of colors by human beings. The following sections on advantages of web-based applications and computer-aided learning serve to provide insight useful for the subsequent chapter, where a concept for the product will be developed.

2.1 Advantages of Web-Based Applications

Web-based applications as learning tools have some obvious potential advantages. Compared to regular, non-web-based software, there is the advantage of portability. Aside from a modern web browser, no additional software or plug-ins need to be installed. Thanks to HTML5 and WebGL, even 3D visualizations like the ones planned for this project can be run in the browser. Fewer requirements in pre-installed software can make an application available to a larger group of people with different operating systems and devices.

Provided the application is optimized for mobile devices, portability is even higher. According to a recent poll, over 90% of Germans who are at least 18 and less than 35 years old reported owning a smartphone [Pou16, p. 20]. Since smartphones are often more readily available than full-sized computers, there are more opportunities for people with smartphones to open a web-based mobile application with the intention to study. The question is whether they will use these opportunities or rather work from a desk. One study from 2012 found that, among surveyed students, about 47% were already using their smartphones for learning while 76% of those who did not would consider it if it benefited their studies [WMN12, p. 8].

Another big advantage of web-based applications and computer programs in general is the ability to present visualizations in 2D or 3D as mentioned above. Especially more complex topics can benefit from interactive rather than static visualizations or from simulations. For example, the original SIMBA module for color theory includes a 3D view of three different color models side-by-side that shows the effects of manipulating any parameter of any of the three models.

2.2 Computer-Aided Learning

Given the advantages of web-based applications presented in Section 2.1, questions of interest are whether computer-aided learning with such tools makes any difference and, if so, what can be done to make them more effective. The first question has been investigated in a number of studies, 23 of which have been analyzed in a meta-analysis [Mea+09, pp. 18f.]. It has been discovered that, compared to face-to-face instruction alone, augmenting this traditional method of studying with the use of online-tools resulted in stronger learning outcomes [Mea+09, p. 19].

In respect to the previously mentioned simulations, the same meta-analysis found their effect to be "modestly positive" [Mea+09, pp. 43f.]. Interestingly, while they did not find quizzes as part of the online tools to be particularly effective [Mea+09, p. 43], asking people to reflect on their progress and understanding did seem to help in the nine studies assessing this method [Mea+09, pp. 44f.].

Mayer [May02, p. 31] defines the *multimedia principle* as people learning "more deeply from words and pictures than from words alone". Furthermore, the so-called *spacial-contiguity effect* describes the notion that people learn more effectively when pictures and corresponding text are placed closely together or when the text is integrated into the illustration [MM99, p. 358]. This effect has been shown in an experiment where text was placed at varying distances to an animation explaining the formation of lightning [MM99, pp. 359f., p. 366]. In practical terms, it is therefore useful to augment text with pictures, especially if the picture and relevant text are not far apart. However, it is also advised not to include pictures that do help explain what is said in the text [May+95, p. 31]. For example, on a webpage teaching about color theory, a picture of a paint brush spanning half the screen might be more distracting rather than conducive to the desired learning outcome.

2.3 Color Models and Color Spaces

Physically, color is made up of visible light, which is electromagnetic radiation of wavelengths between about 380nm and 700nm [10]. Pure spectral colors, known as the colors of the rainbow, are composed of only one wavelength of light. All other colors including white and pink, for example, must therefore be a mixture of several wavelengths. Physiologically, human beings with normal color vision perceive color with three different kinds of cone cells in their eyes. These cones differ in sensitivity to certain wavelengths, peaking at about 419nm, 531nm, and 558nm [DBM83, p. 121], which correspond roughly to blue, green and red.

In the literature, the terms color model, color space, and color system are sometimes used interchangeably [AKM07, p. 1345]. In a more precise definition, a color model describes an abstract way in which colors can be represented via a number of components, for example red, green, and blue. By this definition, a system can only be called a color space if such a model is combined with a definition for how

to interpret the components [GRR10, p. 394]. For instance, in the example of a color model with components for red, green, and blue, one color space may specify a green value equal to 1 with all other components equal to 0 to be slightly more saturated and slightly more yellow than another. The term color system will be used synonymously for both color model and color space in this thesis. If not otherwise specified, component parameters will always be considered to be in the [0,1] interval.

The following sections will provide an overview over a selection of important color systems. For each system, at least one method of conversion to and from another color system will be given. These respective other systems are chosen to eventually allow the conversion between any two of the presented color systems. Starting from an arbitrary color space (with the exception of CIE xy and Munsell), Figure 2.1 shows which conversions are necessary to arrive at the desired other color space.

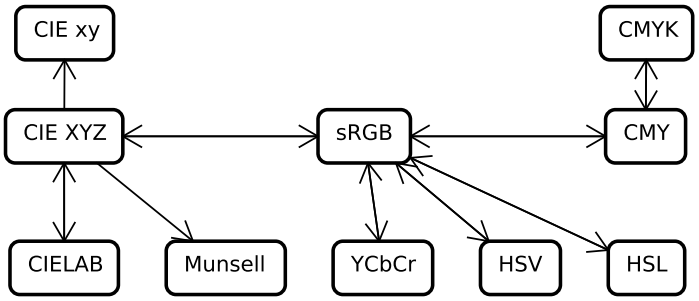


Figure 2.1: Directed graph illustrating the paths necessary to convert between the presented color spaces

2.3.1 CIE 1931 XYZ, CIE 1931 xy, and Gamuts

To compare different color spaces, it is useful to look at their respective gamuts. The gamut of a color space is the set of colors it can represent [BB09b, p. 102]. Gamuts are often visualized using so-called horseshoe or chromaticity diagrams as shown in Figure 2.2, which are a projection of the gamut onto the International Commission on Illumination's (Commission Internationale de l'Eclairage, CIE) x, y coordinate system. The curved line around the horseshoe represents all monochromatic (spectral) colors visible to the human eye. Every other visible color is a combination of at least two different wavelengths and lies within the enclosed shape.

In order to understand this projection, it is necessary to consider the higher-dimensional XYZ color space, which is based on color matching experiments published independently by William D. Wright in 1929 and John Guild in 1931. Subjects were asked to recreate colors of the spectrum by adjusting the intensities of three primary colors [Wri29, pp. 142, 148; Gui32, p. 156]. Wright and Guild first

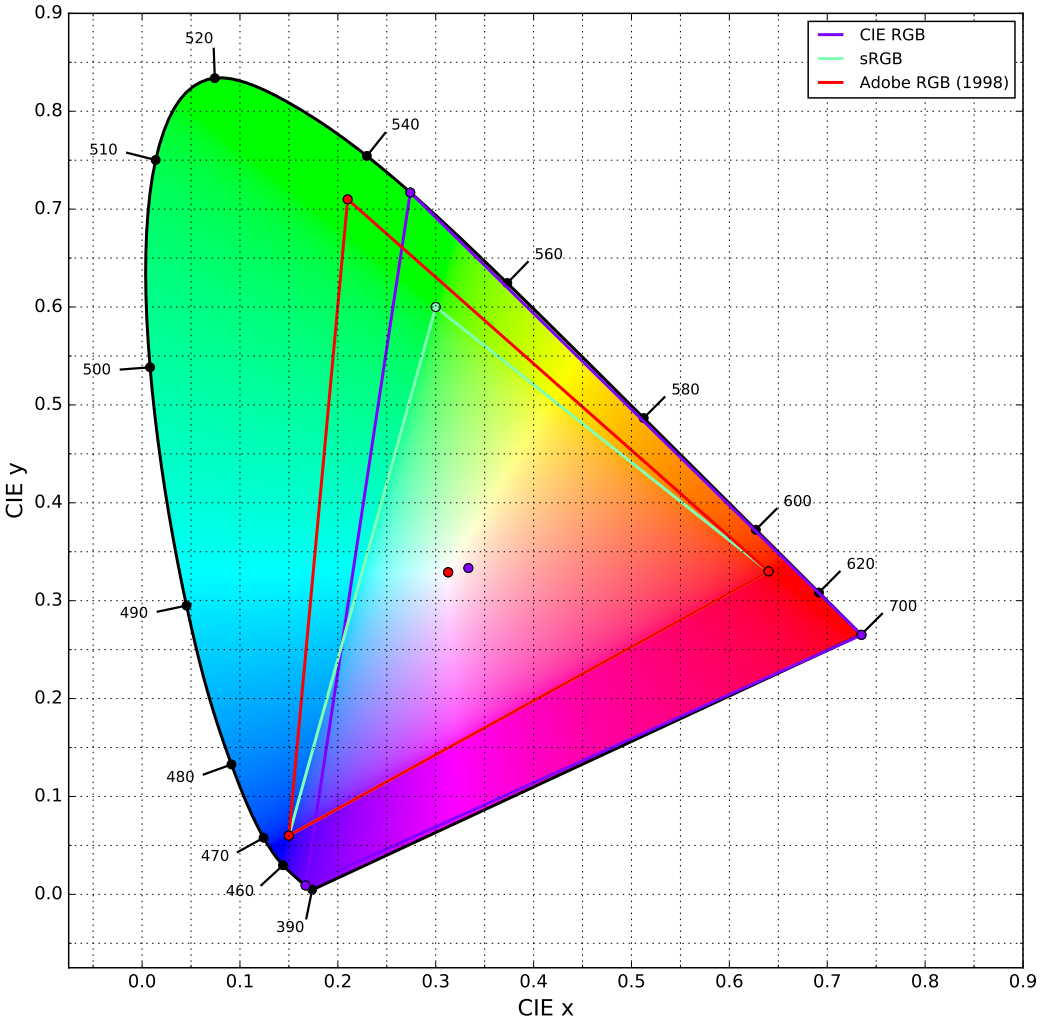


Figure 2.2: Gamut comparison of the CIE RGB, sRGB, and Adobe RGB (1998) color spaces in a CIE x, y chromaticity diagram. Numbers around the curved line denote wavelengths of light in nm. Please note that the colors in this diagram cannot be accurate and are for orientation purposes only. Generated using the Python module *colour* [Man+16].

mapped the data obtained from these experiments using coordinate systems each axis of which corresponded to the intensity of one of the primary colors [Wri29, p. 151; Gui32, p. 164]. However, negative values were possible because no three primaries in combination alone can ever represent all colors of the spectrum and because the colorimeters allowed for compensation if this was the case [Gui32, p. 152].

The experiments mentioned above, which were conducted on several observers, provided the combined data to define one standard observer [SG31, p. 79]. In addition, a particular XYZ color space also depends on an illuminant or white point [BB09b, pp. 101f.]. It can be useful to use different illuminants for different conditions, as colors will be perceived differently in bright sunlight than in the light of an incandescent bulb [SG31, p. 83].

The CIE XYZ color space is a transformation of a three-dimensional coordinate system based on standardized primary colors in such a way as to guarantee the following properties [BB09b, pp. 98f.]:

- For every color $(X, Y, Z)^\mathsf{T}$ visible to human beings, X, Y, and Z are positive.
- Y aligns with the perceived brightness of a color.

On the basis of this XYZ space, one can arrive at an x, y chromaticity diagram as mentioned above via the CIE's definition of the x, y, z coordinates for a point $P = (x', y', z')_{\mathrm{XYZ}} = (x', y', z')^\mathsf{T}$ in the XYZ space [SG31, p. 89]:

$$\frac{x}{x'} = \frac{y}{y'} = \frac{z}{z'} = \frac{1}{x' + y' + z'} \tag{2.1}$$

Therefore it holds that

$$x = \frac{x'}{x' + y' + z'}, \quad y = \frac{y'}{x' + y' + z'}, \quad z = \frac{z'}{x' + y' + z'}. \tag{2.2}$$

Any such point P can be described as a line $l = 0 + \lambda \cdot P$ through the origin and P. From Equation 2.2 it follows that $x + y + z = 1$ and that every P is scaled by $\lambda = \frac{1}{x'+y'+z'}$. Therefore, Equation 2.1 describes a projection through the origin onto the plane $\tilde{x} + \tilde{y} + \tilde{z} = 1$ for XYZ points $(\tilde{x}, \tilde{y}, \tilde{z})^\mathsf{T}$. The CIE x, y coordinate system can then be attained simply by ignoring the z component [BB09b, pp. 99ff.].

2.3.2 RGB and sRGB

As in Wright's and Guild's experiments described in Section 2.3.1, RGB colors are produced by adding different intensities of red, green, and blue light. The RGB color model can be understood as a cube in a three-dimensional Cartesian coordinate system with one axis each for the red, green, and blue channels [BB09a, p. 186]. A color can therefore be defined as a vector $(r, g, b)^\mathsf{T}$ or a tuple $(r, g, b)_{\mathrm{RGB}}$,

where r, g, and b are the color's intensities for each channel. Figure 2.3 shows the RGB cube for channel values in the $[0, 1]$ interval. However, in computer graphics, integral values between 0 and 255 (between 0 and FF in the hexadecimal system) are also a popular choice. For example, RGB colors can be specified in either notation both in CSS [Ç+11] and in OpenGL up to version 3.1 [06]. (Immediate-mode vertex attribute specification was deprecated with OpenGL 3.0 [Ope16].)

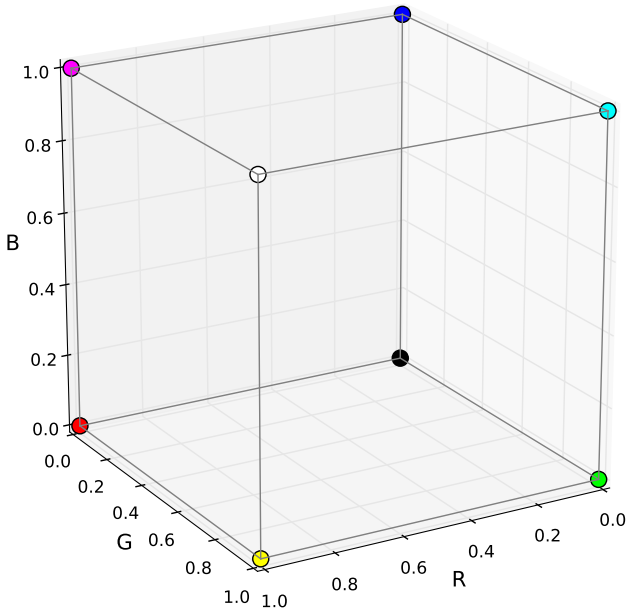


Figure 2.3: The RGB model represented as a cube: Vertices are colored according to the RGB color corresponding to their position within the coordinate system [BB09a, p. 186, adapted].

Since the definition of the RGB model does not specify the exact shades of the three primaries red, green, and blue, different RGB color spaces with different gamuts are possible. Some examples are plotted in Figure 2.2, which shows the CIE RGB, standard RGB (sRGB), and Adobe (1998) color spaces. Among other parameters, sRGB defines locations for the three primaries red, green, and blue within the CIE x, y space (see Table 2.1), as well as a transfer function for gamma correction [13, pp. 1f.]. The transfer function for each component $c_{\text{lin}} \in \{r, g, b\}$ of an RGB color $(r, g, b)^{\mathsf{T}}$ with linear intensities is shown in the following equation.

$$c = \begin{cases} 12.92 \cdot c_{\text{lin}} & \text{if } c_{\text{lin}} \leq 0.0031308 \\ 1.055 \cdot c_{\text{lin}}^{\frac{1}{2.4}} - 0.055 & \text{if } c_{\text{lin}} > 0.0031308 \end{cases} \tag{2.3}$$

Table 2.1: Specification of the sRGB primary colors within the CIE 1931 x, y color space [13, p. 1]

Primary	**x**	**y**
R	0.64	0.33
G	0.30	0.60
B	0.15	0.06

In order to linearize a gamma-corrected RGB color, Equation 2.3 can be reversed:

$$c_{\text{lin}} = \begin{cases} \frac{c}{12.92} & \text{if } c \leq 0.0031308 \cdot 12.92 \\ \left(\frac{c+0.055}{1.055}\right)^{2.4} & \text{if } c > 0.0031308 \cdot 12.92 \end{cases} \tag{2.4}$$

Linear RGB colors $(r_{\text{lin}}, g_{\text{lin}}, b_{\text{lin}})^{\mathsf{T}}$ can be converted from CIE XYZ coordinates $(x', y', z')^{\mathsf{T}}$, given the standard illuminant D65, via the transformation in Equation 2.5 [13, p. 1]. Given a linearized RGB color, the XYZ coordinates can be calculated with the inverse matrix as in Equation 2.6.

$$\begin{aligned} \begin{pmatrix} r_{\text{lin}} \\ g_{\text{lin}} \\ b_{\text{lin}} \end{pmatrix} &= M \cdot \begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} \\ &= \begin{bmatrix} 3.2406255 & -1.537208 & -0.4986286 \\ -0.9689307 & 1.8757561 & 0.0415175 \\ 0.0557101 & -0.2040211 & 1.0569959 \end{bmatrix} \cdot \begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} \end{aligned} \tag{2.5}$$

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = M^{-1} \cdot \begin{pmatrix} r_{\text{lin}} \\ g_{\text{lin}} \\ b_{\text{lin}} \end{pmatrix} = \begin{bmatrix} 0.4124 & 0.3576 & 0.1805 \\ 0.2126 & 0.7152 & 0.0722 \\ 0.0193 & 0.1192 & 0.9505 \end{bmatrix} \cdot \begin{pmatrix} r_{\text{lin}} \\ g_{\text{lin}} \\ b_{\text{lin}} \end{pmatrix} \tag{2.6}$$

2.3.3 CMY and CMYK

Unlike RGB (see Section 2.3.2), the CMY color model makes use of subtractive color mixing with the primaries cyan, magenta, and yellow, rather than additive color mixing. Subtractive mixing may be familiar to most from painting with watercolors. Furthermore, pigments for the colors cyan, magenta, and yellow are commonly used in printers.

In principle, converting colors from RGB to CMY and back is simple. In the RGB model, cyan can be obtained from a mixture of only green and blue, which means when used as a filter or pigment, cyan absorbs red light. Similarly, magenta absorbs green and yellow absorbs blue light. Using this information, the conversion between an RGB color $(r, g, b)^{\mathsf{T}}$ and the corresponding CMY color $(c, m, y)_{\text{CMY}} = (c, m, y)^{\mathsf{T}}$ works by starting with white and subtracting the color in the opposite

representation [BB09a, p. 223], as given in the following equations.

$$\begin{pmatrix} c \\ m \\ y \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} - \begin{pmatrix} r \\ g \\ b \end{pmatrix} \tag{2.7}$$

$$\begin{pmatrix} r \\ g \\ b \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} - \begin{pmatrix} c \\ m \\ y \end{pmatrix} \tag{2.8}$$

However, there is a problem with this method when used in practice. According to Equation 2.7, black is the mixture of 100% cyan, magenta, and yellow. As can also be experienced with watercolors though, cleaning a brush in a glass of water after painting with different colors more easily turns the water into a brown hue than actual black. The common solution for this problem is to introduce a fourth, black channel K. Another problem is the fact that CMYK and color models like RGB are designed for different kinds of devices which often operate with different gamuts. Conversion to CMYK is nontrivial and device-dependent [HSF06, p. 159; BB09a, p. 224]. For the scope of this thesis, only the straight-forward method for converting CMY colors $(c, m, y)^\mathsf{T}$ to CMYK colors $(c', m', y', k)_{\mathrm{CMYK}} = (c', m', y', k)^\mathsf{T}$ will be considered. That is

$$\begin{pmatrix} c' \\ m' \\ y' \\ k \end{pmatrix} = \begin{pmatrix} c - k \\ m - k \\ y - k \\ k \end{pmatrix}, \tag{2.9}$$

where $k = \min\{c, m, y\}$ [BB09a, p. 223]. Because of the introduction of the K channel, less intensity is needed in the other three channels. This is accounted for by the subtraction of k from the original CMY color in Equation 2.9.

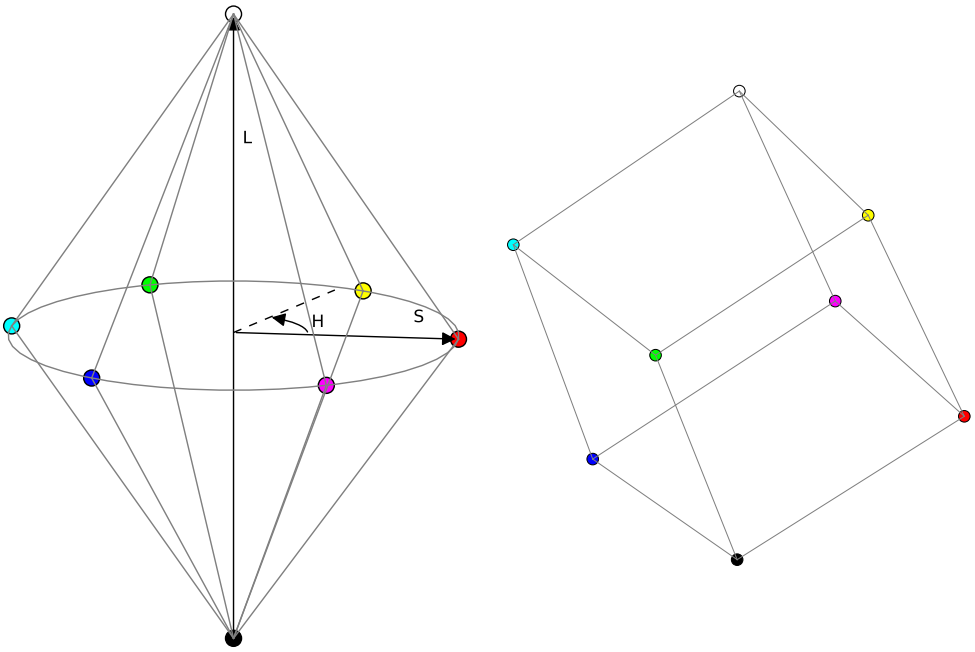
2.3.4 HSL

Even though most human eyes detect colors similarly to how the RGB model works (see Section 2.3), we normally do not think or talk about colors as a mixture of these three components. We might, however, talk of colors as being more or less saturated than others, having different hues or tones, or as being brighter than others. For this reason, many software applications (open-source examples are Blender[1], the GNU Image Manipulation Program[2] (GIMP), Inkscape[3], and Krita[4]) include color pickers which try to cater to our perception of colors in the terms mentioned above [JG78, p. 21].

One such perceptual color model is HSL (also called HLS [BB09a, p. 207]), which

[1] `https://www.blender.org/`
[2] `https://www.gimp.org/`
[3] `https://inkscape.org/en/`
[4] `https://krita.org/en/`

is an acronym for hue, saturation, and lightness. Originally, it was introduced by Joblove and Greenberg as "hue/chroma/intensity" [JG78, pp. 22f.]. They describe the color space as a biconal solid similar to Figure 2.4a, in which the vertical axis represents all shades of gray between 0 (black) and 1 (white). All fully saturated colors then lie on the outer circle of the common basis of both cones at $L = 0.5$, which allows for the hue to be defined as an angle. The third parameter, saturation, corresponds to the radius of the circle around the vertical axis at the position of the current lightness.



(a) The HSL color space in its biconal representation [BB09a, p. 207, adapted]

(b) The rotated RGB cube

Figure 2.4: Comparison of the HSL cones with the rotated RGB cube

Given an RGB color $(r, g, b)^\mathsf{T}$, the lightness l, and saturation s_{HSL} of an HSL color $(h, s_{\text{HSL}}, l)_{\text{HSL}} = (h, s_{\text{HSL}}, l)^\mathsf{T}$ are defined as follows [BB09a, pp. 212f.]:

$$l = \frac{\max\{r, g, b\} + \min\{r, g, b\}}{2} \tag{2.10}$$

$$s_{\text{HSL}} = \begin{cases} 0 & \text{if } l = 0 \\ 0.5 \cdot \frac{\max\{r,g,b\} - \min\{r,g,b\}}{l} & \text{if } 0 < l \leq 0.5 \\ 0.5 \cdot \frac{\max\{r,g,b\} - \min\{r,g,b\}}{1-l} & \text{if } 0.5 < l < 1 \\ 0 & \text{if } l = 1 \end{cases} \tag{2.11}$$

The reasoning for these conversion formulae can be better understood when visualizing the RGB cube from Figure 2.3 tilted so that black is still in the origin

and the diagonal line where $r = g = b$ now aligns with the vertical axis. For the lightness component, Equation 2.10 relates the original RGB color to said diagonal within the RGB cube, while Equation 2.11 establishes a measure of distance from this diagonal line for the saturation.

The hue $h \in [0, 1]$ corresponds to an angle $h \cdot 2\pi$ in radians. It is calculated from the given RGB color with intermediate steps as follows [BB09a, p. 208]:

$$\begin{pmatrix} r' \\ g' \\ b' \end{pmatrix} = \begin{pmatrix} \frac{\max\{r,g,b\}-r}{\max\{r,g,b\}-\min\{r,g,b\}} \\ \frac{\max\{r,g,b\}-g}{\max\{r,g,b\}-\min\{r,g,b\}} \\ \frac{\max\{r,g,b\}-b}{\max\{r,g,b\}-\min\{r,g,b\}} \end{pmatrix} \tag{2.12}$$

$$h' = \begin{cases} b' - g' & \text{if } \max\{r, g, b\} = r \\ r' - b' + 2 & \text{if } \max\{r, g, b\} = g \\ g' - r' + 4 & \text{if } \max\{r, g, b\} = b \end{cases} \tag{2.13}$$

$$h = \frac{h'}{6} \mod 1 \tag{2.14}$$

Because $r, g, b \in [0, 1]$, r', g', and b' are all contained within the interval $[0, 1]$. Without loss of generality, this can be seen if $\max\{r, g, b\}$ is assumed to be r. In that case, r' is equal to 0 and $g' = \frac{r-g}{r-\min\{r,g,b\}}$, $b' = \frac{r-b}{r-\min\{r,g,b\}} \in [0, 1]$, since the numerator can only be less than or equal to the denominator. Therefore, each of the sums in the calculation of h' must be in the interval $[-1, 1]$. Consequently, $h' \in [-1, 5]$ and $h \in [0, 1]$.

In the special cases of fully saturated red, green, and blue, the conversion results in hue values of 0, $\frac{2}{6}$, and $\frac{4}{6}$, respectively. The other non-gray vertices of the RGB cube yellow, cyan, and magenta are assigned the hues $\frac{1}{6}$, $\frac{3}{6}$, and $\frac{5}{6}$. Note that, in the case of yellow where $\max\{r, g, b\} = r = g$, it makes no difference which of the two fitting cases is calculated.

To convert an HSL color $(h, s_{\text{HSL}}, l)^\mathsf{T}$ back into the RGB color $(r, g, b)^\mathsf{T}$, first the two cases need to be considered where hue and saturation are undefined [BB09a, p. 213]:

$$\begin{pmatrix} r \\ g \\ b \end{pmatrix} = \begin{cases} (0, 0, 0)^\mathsf{T} & \text{if } l = 0 \\ (1, 1, 1)^\mathsf{T} & \text{if } l = 1 \end{cases} \tag{2.15}$$

Equation 2.13 partitions the circle of hues into three times two sectors, depending on which component in the original RGB color was larger than the others. In order to reverse this and to obtain the RGB color, it is helpful to first calculate some intermediate values [BB09a, p. 214] as follows:

$$\begin{aligned} h' &= (6 \cdot h) \mod 6 \\ c_1 &= \lfloor h' \rfloor \\ c_2 &= h' - c_1 \end{aligned}$$

$$
\begin{aligned}
d &= \begin{cases} s_{\text{HSL}} \cdot l & \text{if } l \leq 0.5 \\ s_{\text{HSL}} \cdot (1 - l) & \text{if } l > 0.5 \end{cases} \\
u_1 &= l + d \\
u_2 &= l - d \\
u_3 &= u_1 - (u_1 - u_2) \cdot c_2 \\
u_4 &= u_2 + (u_1 - u_2) \cdot c_2
\end{aligned} \tag{2.16}
$$

Here, c_1 is an index for the hue's sector and c_2 is the position in that sector. As s_{HSL} models a radius between 0 and 1, d can be understood as the radius adjusted for the biconal representation of the HSL color space. It is guaranteed to be within the interval $[0, 0.5]$. Therefore, u_1 will always be the maximum component in the new RGB color, and u_2 will be the minimum. u_3 and u_4 interpolate between the maximum and minimum with varying hues in different directions. From these intermediate values, the final RGB values can be assembled [BB09a, p. 214]:

$$
\begin{pmatrix} r \\ g \\ b \end{pmatrix} = \begin{cases} (u_1, u_4, u_2)^\mathsf{T} & \text{if } c_1 = 0 \\ (u_3, u_1, u_2)^\mathsf{T} & \text{if } c_1 = 1 \\ (u_2, u_1, u_4)^\mathsf{T} & \text{if } c_1 = 2 \\ (u_2, u_3, u_1)^\mathsf{T} & \text{if } c_1 = 3 \\ (u_4, u_2, u_1)^\mathsf{T} & \text{if } c_1 = 4 \\ (u_1, u_2, u_3)^\mathsf{T} & \text{if } c_1 = 5 \end{cases} \tag{2.17}
$$

While the HSL color model has been designed to make color-picking easier for human beings, it is not perfect. One reason for this is the assignment of the same lightness to all three primaries. As seen in Table 2.1, in the sRGB space primaries are assigned different positions on the y axis. When transforming these definitions of primary red, green, and blue back into the XYZ space, differences in the Y component remain. Therefore, the primaries as defined for the sRGB space do not have the same perceived brightness, which means lightness in the HSL model is not the same as perceived brightness. This becomes apparent when comparing the HSL colors of maximum saturation and lightness $\frac{1}{2}$ with a gray-scale image of the true perceived brightness values as shown in Figure 2.5.

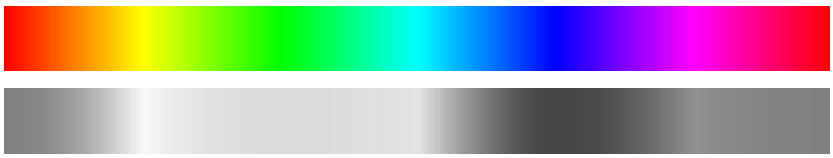


Figure 2.5: Differences in brightness for equal HSL lightness. Top: Colors $(h, 1, \frac{1}{2})_{\text{HSL}} \; \forall \; h \in [0, 1]$. Bottom: Gray-scale colors produced by keeping the Y component in XYZ space constant while removing all chromaticity.

2.3.5 HSV

The HSV color model shares most of its properties with HSL. The letter V stands for value and is sometimes interchanged with the letter B for brightness [BB09a, p. 205], which points to the fact that the third component is defined differently in this color model. Instead of transforming the RGB cube into a biconal shape, HSV starts with a single inverted cone as shown in Figure 2.6. The hue is again an angle which can be calculated in the same way as for HSL with Equations 2.12, 2.13, and 2.14. With the value or brightness for an RGB color $c_{\text{RGB}} = (r, g, b)^{\mathsf{T}}$ now being defined simply as $v = \max\{r, g, b\}$, the saturation s_{HSV} is defined as follows for $c_{\text{RGB}} \neq (0, 0, 0)^{\mathsf{T}}$ [JG78, p. 22]:

$$s_{\text{HSV}} = \frac{\max\{r, g, b\} - \min\{r, g, b\}}{\max\{r, g, b\}} \tag{2.18}$$

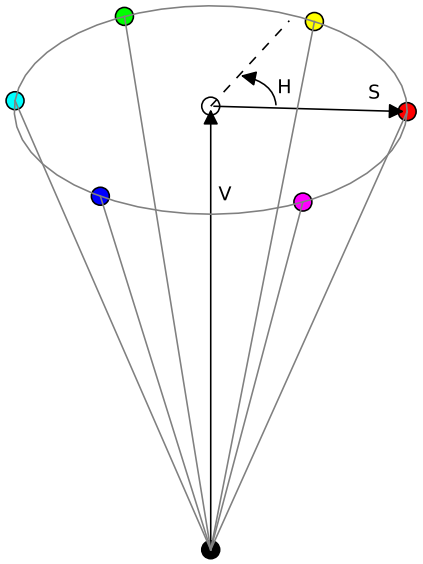


Figure 2.6: The HSV color space represented as a cone [BB09a, p. 207, adapted]

Calculating the RGB values $(r, g, b)^{\mathsf{T}}$ from a given HSV color $(h, s_{\text{HSV}}, v)_{\text{HSV}} = (h, s_{\text{HSV}}, v)^{\mathsf{T}}$ works similarly as it does for HSL. h', c_1, and c_2 are obtained via the formulae given in Equations 2.16. The RGB values are then assembled as follows [BB09a, p. 210]:

$$\begin{aligned} w_1 &= (1 - s_{\text{HSV}}) \cdot v \\ w_2 &= (1 - s_{\text{HSV}} \cdot c_2) \cdot v \\ w_3 &= (1 - s_{\text{HSV}} \cdot (1 - c_2)) \cdot v \end{aligned}$$

$$\begin{pmatrix} r \\ g \\ b \end{pmatrix} = \begin{cases} (v, w_3, w_1)^\mathsf{T} & \text{if } c_1 = 0 \\ (w_2, v, w_1)^\mathsf{T} & \text{if } c_1 = 1 \\ (w_1, v, w_3)^\mathsf{T} & \text{if } c_1 = 2 \\ (w_1, w_2, v)^\mathsf{T} & \text{if } c_1 = 3 \\ (w_3, w_1, v)^\mathsf{T} & \text{if } c_1 = 4 \\ (v, w_1, w_2)^\mathsf{T} & \text{if } c_1 = 5 \end{cases} \tag{2.19}$$

For the same reason described at the end of Section 2.3.4, the value or brightness of an HSV color does not correspond perfectly to a color's brightness as it is perceived by human beings. This means, if the value component is left constant and only saturation or hue are changed, one cannot expect the original and the new color to be perceived as equally bright. Therefore, both value and lightness may at best serve as an approximation of brightness.

2.3.6 Munsell

One solution to the issues with HSL and HSV presented in Section 2.3.4 and Section 2.3.5 is to look at a color system that, at least in part, inspired the creation of the HSL color model [JG78, p. 22]. Like the XYZ color space, but with different tools, the Munsell system is based on experimental data [Mun12, p. 236].

The system describes a so-called "color-tree" [Mun12, p. 240]. Similar to HSL and HSV, there is a vertical axis representing brightness, in this case also called value. Here, the human eye's response to light intensity is modeled logarithmically [Mun12, p. 237]. Hue is again defined as an angle with complementary colors positioned opposite to each other, and saturation, in this case called chroma, is defined as a radius [Mun12, pp. 237f.]. An important difference to HSL and HSV is the following: Because the Munsell chroma is based strictly on human perception of color saturation, the color space cannot be represented in a perfectly round shape [Mun12, p. 240]. Rather, from a top-down view, clear protrusions are visible for example around the hue angle for red and for a mixture of blue and purple [Mun12, p. 242], which indicates that pure colors of these hues are perceived as more saturated than others.

Munsell recorded samples within this color space in his Munsell Book of Color. In 1943, with the intention to reduce irregularities in the spacing of those color samples, the samples were analyzed experimentally and their position within the CIE xyY color space was determined [NNJ43, pp. 385f.]. This data can be used to convert colors from the Munsell system into xyY.

2.3.7 CIELAB

Another popular solution for the issues with HSL and HSV presented in Section 2.3.4 and Section 2.3.5 is the CIE L*a*b* color space. One of the goals

for developing this color space was a formula to quantify the perceived difference between two colors [McL76, pp. 339f.].

The L*a*b* space is defined as a transformation of the CIE 1931 XYZ color space via the following equations for L*a*b* colors $(l, a, b)^\mathsf{T}$ and XYZ colors $(x', y', z')^\mathsf{T}$ [McL76, p. 340]:

$$\begin{aligned} l &= 117 \cdot f\left(\frac{y'}{Y_0}\right) - 16 \\ a &= 500 \cdot \left(f\left(\frac{x'}{X_0}\right) - f\left(\frac{y'}{Y_0}\right)\right) \\ b &= 200 \cdot \left(f\left(\frac{y'}{Y_0}\right) - f\left(\frac{z'}{Z_0}\right)\right), \end{aligned} \tag{2.20}$$

where (X_0, Y_0, Z_0) is a given illuminant or white point, typically the standard illuminant D65 [BB09b, p. 104]. D65 was determined to correspond roughly to daylight with a color temperature of 6500°K and has the xyz coordinates $(0.3127, 0.3291, 0.3582)^\mathsf{T}$, given the CIE 1931 standard observer [Jud+64, p. 1036]. Normalizing to $Y = 1$ yields the approximate XYZ coordinates $(0.9502, 1.0, 1.0884)^\mathsf{T}$. The function f was originally defined as $f(c) = c^{\frac{1}{3}}$ if $c > c_t = 0.01$. Otherwise the so-called ANLAB equation was supposed to be used in order to avoid negative L* values for dark colors close to $Y = 0$ [McL76, p. 340]. For the ISO 31655 standard, this has been slightly changed to the following function f_1 with $c_{t_1} = 0.008856$ [BB09b, p. 104]:

$$f_1(c) = \begin{cases} c^{\frac{1}{3}} & \text{if } c > c_{t_1} \\ 7.787 \cdot c + \frac{16}{116} & \text{if } c \leq c_{t_1} \end{cases} \tag{2.21}$$

Converting from a given L*a*b* color back into XYZ space works as follows [BB09b, p. 105]:

$$\begin{aligned} f_2(c) &= \begin{cases} c^3 & \text{if } c^3 > c_{t_1} \\ \frac{c - \frac{16}{116}}{7.787} & \text{if } c^3 \leq c_{t_1} \end{cases} \\ y'_0 &= \frac{l + 16}{116} \\ x' &= X_0 \cdot f_2\left(\frac{a}{500} + y'_0\right) \\ y' &= Y_0 \cdot f_2\left(y'_0\right) \\ z' &= Z_0 \cdot f_2\left(y'_0 - \frac{b}{200}\right) \end{aligned} \tag{2.22}$$

Due to the design of this color space to be uniform, the difference between two colors $(l_1, a_1, b_1)^\mathsf{T}$ and $(l_2, a_2, b_2)^\mathsf{T}$ will be perceived approximately the same as the difference between two other colors if the Euclidean distance in L*a*b* space is the same. The square of the Euclidean distance, shown in Equation 2.23, is called

the color-difference formula [McL76, p. 340].

$$\Delta E_{(\text{CIELAB})} = (l_2 - l_1)^2 + (a_2 - a_1)^2 + (b_2 - b_1)^2 \tag{2.23}$$

2.3.8 YCbCr

The YCbCr model is of interest due to its use in television and the JPEG image compression. Y represents luminance, C_b is the weighted difference between the luminance and blue, and, similarly, C_r is the weighted difference between the luminance and red. The general conversion from an RGB color $(r, g, b)^\mathsf{T}$ into YCbCr represented as $(y, c_b, c_r)^\mathsf{T}$ is defined as follows for weights w_R, w_G, and $w_B = 1 - w_R - w_G$ [BB09b, p. 221]:

$$\begin{aligned} y &= w_R \cdot r + w_G \cdot g + w_B \cdot b \\ c_b &= \frac{1}{2 \cdot (1 - w_B)} \cdot (b - y) \\ c_r &= \frac{1}{2 \cdot (1 - w_R)} \cdot (r - y) \end{aligned} \tag{2.24}$$

The direction from YCbCr to RGB is given as follows:

$$\begin{aligned} r &= y + 2 \cdot (1 - w_R) \cdot c_r \\ g &= y - 2 \cdot \frac{w_B \cdot (1 - w_B) \cdot c_b - w_R \cdot (1 - w_R) \cdot c_r}{w_G} \\ b &= y + 2 \cdot (1 - w_B) \cdot c_b \end{aligned} \tag{2.25}$$

Alternatively, once the three weights are given, the Equations 2.24 can be written as a transformation matrix. It is then possible to use the inverse matrix for conversion in the other direction. Table 2.2 shows a list of recommended weights for different use cases.

Table 2.2: Examples of weight parameters for YCbCr-RGB conversion and their recommended use.

Recommended use	$\mathbf{w_r}$	$\mathbf{w_g}$	$\mathbf{w_b}$
Standard television [11a, p. 2], JPEG [11b, pp. 3f.]	0.299	0.587	0.114
HD television [15b, p. 4]	0.2126	0.7152	0.0722
UHD television [15a, p. 4]	0.2627	0.6780	0.0593

3 Concept

This chapter first presents a concept for the product at large in Section 3.1. The two central components of this concept then are discussed in more detail in Section 3.2 and Section 3.3.

3.1 Analysis of Important Components

As shown in Figure 3.1, generally speaking, the purpose of the exercise that is to be developed is threefold. Firstly, it provides the student with opportunities to learn about color and color vision in general. Secondly, the student is able to learn about specific color models and color spaces, and, finally, there is functionality for the student to practice what they already know or what they learned in the first two cases. In the context of this thesis, especially the second and third use case are of importance. The second use case about color models and color spaces reflects the contents of Section 2.3 and includes interactive visualizations of individual color spaces. The third use case can be used by students to study for an exam, or by any person to reflect on what they already understand and where further study might be required.

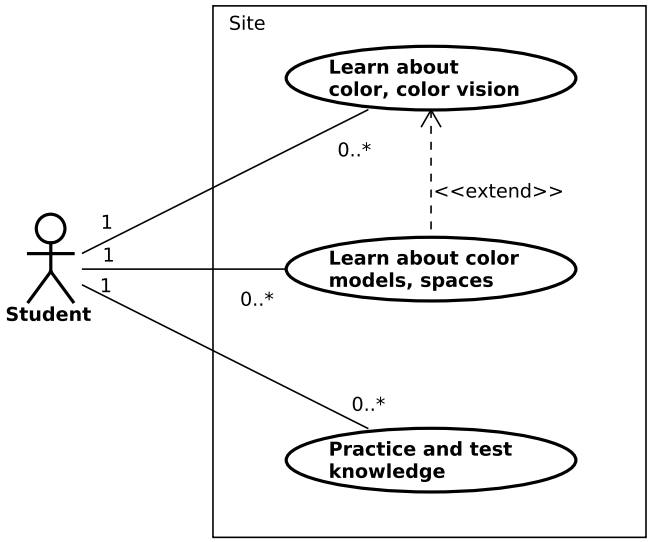


Figure 3.1: Use case diagram of the basic tasks a person using the product might wish to accomplish

Each color model or color space is presented on its own page. Figure 3.2 shows how these pages are related to the interactive visualizations. In addition to plain HTML text and links, a page may include any number of figures. Visualizations are treated as a special kind of figure and visualize at least one color space. The case in which one visualization shows more than one color space can be useful for comparing different color spaces or for visualizing conversions.

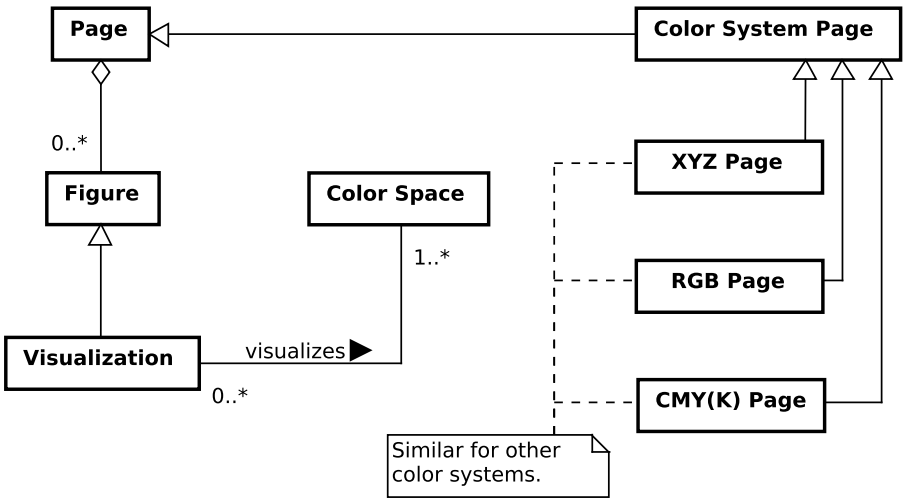


Figure 3.2: Class diagram showing the composition of a page for learning about color models and color spaces

Interaction with a visualization depends on the color spaces currently shown. Figure 3.3 shows a sketch of what the controls for such a visualization might look like when the RGB cube and the HSL space are visualized side-by-side. Below each three-dimensional representation are sliders and input boxes for adjusting the relevant parameters of each system. Changing any one of the sliders triggers the following changes: The selected color in the bottom left is updated, the location of the selected color within the respective color spaces is shown, and the inputs of the other color models are adjusted to show the parameters for the same selected color. So far in this thesis, most color spaces have been drawn as a wireframe model. In the lower right corner of Figure 3.3 is shown a button for revealing a set of advanced controls. These may include settings for switching between a wireframe representation and an opaque or half-transparent solid. Additionally, options for switching between floating point RGB values and values between 0 and 255 may be implemented, as well as the option to represent hue in the [0,1] interval, in radians, degrees, or as integers between 0 and 255 as well. Since these settings are optional and not immediately relevant to understanding the color models, they are hidden by default.

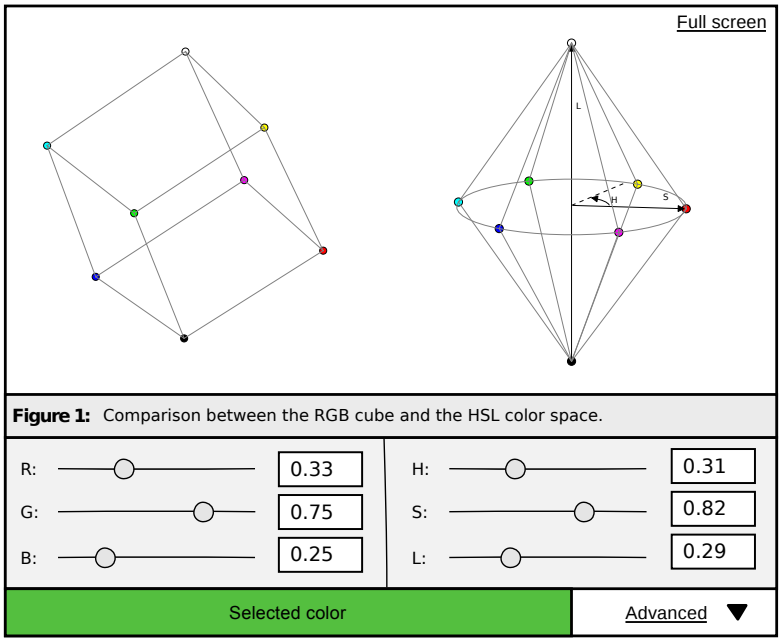


Figure 3.3: UI sketch of an interactive visualization and comparison of RGB and HSL

3.2 Visualizations

In the following, potential visualizations for each of the presented color systems in Section 2.3 will be discussed.

CIE XYZ, CIE xy: A chromaticity diagram in CIE xy space as shown in Figure 2.2 is essential because it helps to illustrate the limits of practical color spaces, as well as differences between them. The transformation from the three-dimensional XYZ space into the two-dimensional x, y space can be demonstrated by showing the intersection of the visible gamut with the $\tilde{x}+\tilde{y}+\tilde{z} = 1$ plane. If the visible gamut is rendered transparent, the resulting projections can be drawn directly onto the projection planes.

RGB and sRGB: Most important for the RGB model is the RGB cube as shown in Figure 2.3 and Figure 3.3. For demonstrating the perceptual non-uniformity of the sRGB space, the visual comparison with another color space is useful. By design, the CIELAB or Munsell system would be ideal. However, because of the relative simplicity of the RGB model, it can be expected that it will be among the first a student will learn. This issue can be addressed by giving the student a list of suggested color spaces for comparison with links to their respective pages. In a visualization similar to Figure 3.3 the student may then select one of the suggested color spaces which they are already familiar with for comparison with the RGB cube.

CMY and CMYK: Since the CMY cube is merely a flipped RGB cube, the visualization of CMY is very similar to that of RGB.

HSL and HSV: Seeing the double cone of Figure 2.4a for HSL or the single cone of Figure 2.6 is useful for understanding the definition of the HSL and HSV color models. The conversion formulae describe a cylinder which is often shown from a top-down view in color pickers, for example in the open-source programs Blender[1] or MyPaint[2]. Another possible representation is an HSL or HSV cube, which is produced by treating the three components of the respective color model as axes for a Cartesian coordinate system. MyPaint, for instance, includes a color picker which allows a user to view the HSV cube from three different sides. Therefore, in order to help students understand the relationship between the original color model and what they may be used to seeing in color pickers, a visual comparison between different representations of HSL and HSV can be useful.

In addition, a re-creation of Figure 2.4, in which HSL and RGB can be compared side-by-side, is useful to illustrate the conversion of colors. As the SIMBA visualizations demonstrate, this comparison is not limited to HSL and RGB. HSV and RGB can be compared in the same way, and a comparison of HSL and HSV can be used to highlight the similarities and the differences between the two color models.

Munsell: Given the Munsell renotation data [NNJ43, pp. 397–405], it is possible to render the Munsell color tree in three dimensions as it is described in Section 2.3.6. As a demonstration of the difference to HSL and HSV, one of these color systems may be displayed in addition to the color tree.

CIELAB: The L*a*b* color space is a transformation of the XYZ space and it, too, contains all colors visible to the human eye. One visualization may therefore show the visible gamut or, for example, the sRGB gamut in three-dimensional space. A visual comparison to the XYZ space can furthermore demonstrate the function of L*a*b*, which is to keep colors that are perceived equally distant from one another equally distant in the color space.

YCbCr: As explained in Section 2.3.8, the YCbCr color space can be arrived at via applying a transformation matrix to any RGB color. Therefore, depending on the parameters w_R, w_G, and w_B, the YCbCr color space can be visualized as an appropriately transformed RGB cube inside a Cartesian coordinate system with axes for Y, C_b, and C_r.

[1] `https://www.blender.org/`
[2] `http://mypaint.org/`

3.3 Exercises

The SIMBA project for computer-generated color already includes three exercises worthy of being reproduced with only minor adjustments. In this section, these exercises will be discussed in regards to their strengths and potential weaknesses. Finally, possible augmentations will be explored and a new exercise will be presented.

3.3.1 Color Matching Exercise

In an instance of what shall be called a color matching exercise, a random but constant color is shown in a rectangle on the left and another adjustable color is shown on the right. The second color can be adjusted via sliders. Using these, the user's task is to get both rectangles to show the same color. If the selected color is close to the original, a label indicates that they are almost correct, and if the two colors match, the label will notify about this as well.

In the SIMBA project, this exercise is only available for the RGB color system. In this context, it provides a good way for students to experience the lack of intuitiveness with using the RGB model for color picking. Moreover, this exercise encourages the user to apply their understanding of the model's parameters. In contrast, if the student were not given the task of actually matching two colors, they would possibly explore predominantly the extrema for each parameter and, in the process, might not fully grasp the influence each parameter has on the final color. For example, when one has tested all eight RGB colors $(0,0,0)^\mathsf{T}, (0,0,1)^\mathsf{T}, \ldots, (1,1,1)^\mathsf{T}$, it is not yet clear where to find a certain shade of brown.

Thanks to its simplicity, this color matching exercise can be easily adapted for any other color system. Almost every model presented in this thesis allows a color to be selected by adjusting three or four sliders. Giving students the possibility of comparing variations of this exercise for different color systems may help them grasp both the meaning of each parameter, as well as advantages and disadvantages of each system in respect to ease of use.

One problem with the original exercise is that, from time to time, the two colors will look as though they are the same, but no indication for a successful match will be given. To reduce frustration on the part of the user, additional indicators may be introduced. For example, the CIE 1976 color difference formula [McL76, p. 340] can be used to quantify the perceptual difference between the two colors. Showing the Euclidean distance of the two colors within the RGB cube might be useful for pointing out that, in the above scenario, the two colors are indeed not the same. On the other hand, one must be careful not to let these indicators tempt students into cheating the exercise. Given the Euclidean distance, for example, each RGB component could be adjusted one after the other to the point where the distance is minimal, not requiring much knowledge about the color space. This can be avoided to some extent by only revealing the distance after the push of a

button.

3.3.2 Color Selection Exercise

The user's interaction with this exercise is shown in an activity diagram in Figure 3.4. Essentially, the task is to find a random color given in numerical form among a list of other random colors. In the SIMBA project, this task is considered easy. Provided the differences between the colors to select from are not too large and the list of colors is not too long, a student should be quick to find the correct color. The original exercise displays seven colors in addition to the target.

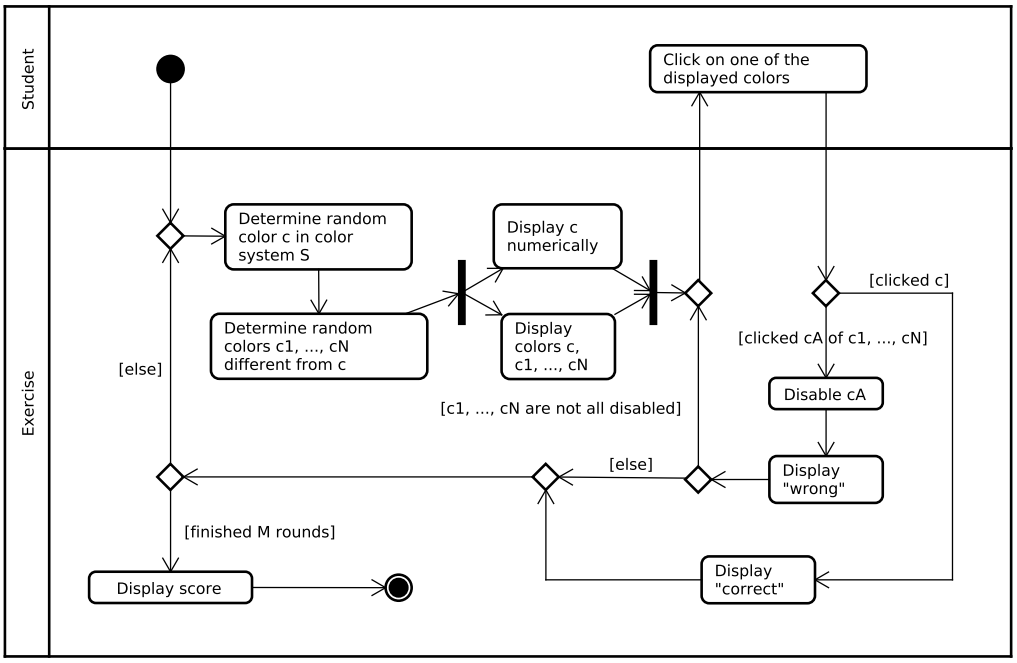


Figure 3.4: Activity diagram for a color selection exercise. In each of M rounds, the student is presented with a random numerical color vector of a color system S. This color is shuffled with N other colors and drawn to the screen. The student must select that color which matches the given numerical representation.

Like the color matching exercise of Section 3.3.1, the color selection exercise is flexible in regards to the color systems it can be used with. Either exactly one color system can be practiced in particular, or one color system of a pre-selected set of systems may be chosen at random in each round for the numerical representation. The latter option is most useful when studying for an exam because it discourages students to, perhaps unconsciously, avoid those color systems with which they are not as familiar with.

The original exercise produces a new set of random colors every time the user

clicks a button. For motivational purposes, however, it may be of advantage to let the student do the exercise for M rounds as shown in Figure 3.4, after which they will receive a score. Ideally, this score will give them an indication for how much they still need to study in order to better understand the color systems in question. To this end, the score may include the number of colors the student selected correctly with the first try. A comparison of their score to the average user may serve as said indicator for self-assessment as well as a motivator for improvement.

3.3.3 Conversion Exercise

For students, a color conversion exercise is especially interesting because it is the easiest to be integrated in exams, which are typically printed in black and white. In SIMBA however, the conversion exercise is available on the push of a checkbox with the label „schwer“ (difficult). The reason for this is that the exercises presented in Section 3.3.1 and Section 3.3.2 tolerate a larger amount of educated guessing. Still, the original exercise does not require the precise calculation of the converted color parameters. For example, when tasked with the conversion of the HSL color $(0.56, 0.6, 0.8)^\mathsf{T}$ into RGB, the answer $(0.67, 0.78, 0.86)^\mathsf{T}$ is considered correct and therefore close enough to the more exact corresponding RGB color $(0.68, 0.83, 0.92)^\mathsf{T}$. In addition, the three-dimensional visualizations of the three color systems RGB, HSL, and HSV are visible the entire time and reflect the user's input. In the following, three possible variations of the conversion exercise will be discussed.

Easy: Only key colors are asked to be converted. For instance, in HSV and HSL, colors that are fully saturated or not saturated at all can be converted between the two systems, as well as from and to RGB, with relative ease.

There are limitations to this variant, however. Key colors that are easy to memorize do not exist in the CIELAB, CIE XYZ, and Munsell color spaces. Furthermore, the different weights recommended for YCbCr complicate the conversion to other color spaces. This variation of the color conversion exercise is therefore limited to RGB, HSL, HSV, and CMY.

Medium: An alternative to the original SIMBA color conversion exercise has been used in an exam before and is somewhat more challenging than the variant Easy. Like in the color selection exercise of Section 3.3.2, students are given a color represented numerically in one color system. Instead of from a list of colored patches, they must now select the matching color from a list of colors represented in another color system. Similar limitations exist as do for the variant Easy, since CIELAB, CIE XYZ, Munsell, and YCbCr can not be easily converted to or from without aid. If HSL or HSV are involved, the task can be slightly simplified further by choosing multiples of $\frac{1}{6}$ or $\frac{1}{12}$ for the hue. In general, too many decimal places should be avoided in order to facilitate

mental arithmetic. Students should focus primarily on understanding the color systems and be encouraged to finish the set number of rounds of the exercise at least once.

Advanced: This variation is closest to the original SIMBA exercise. The user must specify the values of the converted color manually using sliders or the keyboard. Like in the original, a threshold is chosen for a maximum distance between the correct conversion and the values the user selected at which the response is still considered correct. As an alteration, if the user's selection is within this threshold but not equal to the exact result, the exact conversion can be shown and, optionally, compared visually to the selection.

For all three variations, a system of rounds to complete can be implemented equivalent to that shown in Figure 3.4. Because this exercise can be difficult, it makes sense to show visualizations of both color spaces involved in the conversion. On the other hand, since typically no such visual aid is given in exams, the user has the option to switch to an exam mode in which visualizations are disabled for this exercise.

3.3.4 Mixed Tasks Exercise

The aim of this new exercise is to convey the purpose and effects of color systems' parameters in a context that is closer to real-world applications than manual color conversion, for example. Figure 3.5 shows the student's interaction with this exercise similarly to Figure 3.4 for M rounds, after which they receive a score with the number of tasks they solved correctly. The diagram indicates that before the exercise starts, the user first selects a number of color systems they want to practice. Alternatively, this selection can be abbreviated through links to recommended exercises from one of the descriptive color system pages. From the non-empty set of selected color systems, one is chosen randomly in each round. Many of the systems presented here can have several different tasks associated with them. Possible tasks for each color model or color space will be explored in the following.

CIE XYZ, CIE xy: Reduce or increase brightness in XYZ. Since the axes in these color systems represent imaginary primary colors, simple tasks for an exercise cannot be easily constructed.

RGB and sRGB: Reduce or increase brightness. For example, in order to increase brightness, it is sufficient to increase the value of either red, green, or blue. Tasks similar to those for HSL and HSV are also possible, although they would make this exercise very similar to the conversion exercise.

CMY and CMYK: Reduce or increase brightness. This task is similar to the RGB task.

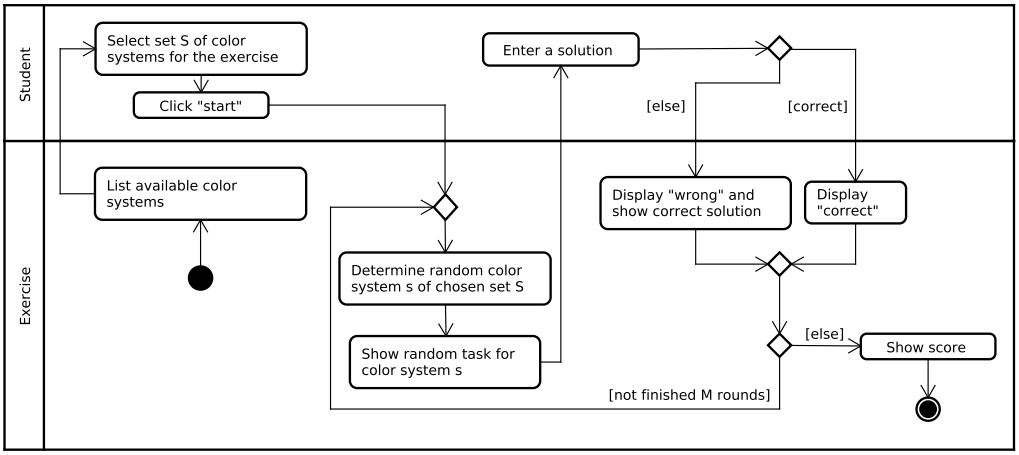


Figure 3.5: Activity diagram for a mixed tasks exercise

HSL and HSV: Given an HSL or HSV color:

- (De-)Saturate (without changing hue or lightness/value).
- Make the color darker/brighter (without changing hue or saturation).
- Give the color a shade between e.g. green and yellow (without changing saturation or lightness/value).

Munsell: Tasks for Munsell can include the tasks for HSL and HSV with adapted terminology.

CIELAB:

- Given two L*a*b* colors x and y, find another color that is as distant from y as y is distant from x.
- Given an L*a*b* color, find another color with less/more chroma. Finding the exact same color with more or less chroma in the CIELAB space is, unfortunately, nontrivial. Especially hues of blue tend to turn purple with increasing distance from the white point and constant brightness [Mor03, pp. 371f.].

YCbCr: Reduce or increase brightness.

In addition to these system-specific tasks, more tasks can be made available for certain combinations of color systems, provided the combination is reflected in the user's selection at the beginning of the exercise. One example is to again ask the user to change the hue of a given color without changing the apparent brightness. Instead of giving this task in the context of the Munsell or CIELAB color space, the student must find the appropriate color system by themselves.

4 Implementation

A selection of the most important implementation details is presented in this chapter. This includes an overview of the tools used to build a website with the desired functionality, as well as descriptions of the implementation of the visualizations and exercises.

4.1 Frameworks, Tools, and Libraries

When building a website, there is a large number of libraries and tools to choose from which aim to make the development process easier. This section gives an overview of the technologies used in this project and explains why they were chosen. The benefits of an implementation in WebGL were already explored in Section 1.3. Using WebGL implies also using HTML and JavaScript.

Javascript / ES6: This project uses the version of JavaScript known by the name of ECMAScript 2015 or ECMAScript 6 (ES6) [Wir15, p. xvii]. Among other things, this version makes object-oriented programming more easily readable by introducing the keyword `class`. Previously, classes had to be written as specialized functions. Another new feature is that classes can be bundled into modules, which, in theory, can be accessed from other classes using the keyword `import`.

Babelify: By the time this project was going to be implemented, the ES6 import feature was not yet supported in modern browsers [16]. However, with the use of a transpiler like Babelify, ES6 can still be used for development. Before publishing, the code is first compiled into a single JavaScript file compatible with current browsers. Babelify is actually a combination of two tools: The first is Babel, which translates all ES6 code but does not resolve dependencies defined via `import`. The latter is done with the help of a tool called Browserify.

Browserify-Shader Like Browserify resolves references to other JavaScript files, Browserify-Shader enables the inclusion of shader text files via the `require` function.

Three.js: Three.js adds an abstraction layer on top of WebGL. It provides scene graph functionality, which allows to organize 3D objects hierarchically, thus making the positioning and animation of elements somewhat simpler. The library also comes with a number of predefined primitives and materials.

Jquery: Jquery is a helpful library which makes it easy to manipulate the HTML content of a page via CSS-like selectors. This functionality can be leveraged, for example, to dynamically insert slider controls depending on the currently shown visualization.

MathJax: Since the conversion between color spaces requires some mathematical formulae, it makes sense to make these as readable as possible. MathJax can interpret LaTeX code embedded in HTML.

UglifyJS: With ES6 and Browserify it is possible to subclass existing Three.js classes. The advantage is that Three.js does not need to be included in the HTML header in its entirety. On the other hand, this leads to a very large new JavaScript file of almost 4MiB. UglifyJS can optimize this code for example by removing redundant whitespaces or renaming functions and variables. It manages to shrink the original file down to only about 870KiB. This file size can be reduced even more by instructing the server to use gzip compression.

Less CSS: Less CSS, an extension of CSS, is used to make the project's CSS code more maintainable. This is achieved with the use of variables, modularization, and nested selectors, among other things. Since browsers only support regular CSS, code written in Less first needs to be compiled before publishing.

Normalize.css: Different browsers may have different default settings for the font or spacing between paragraphs, for example. Normalize.css aims to make websites look the same regardless of the browser used.

PHP: The evaluation of this project requires server-side scripting in order to keep the experimental groups separate. For server-side scripting, PHP is a popular choice and, importantly, is supported by most hosting providers.

Symfony: Symfony is a PHP framework that includes useful features like templating via the templating engine Twig, as well as object-relational mapping (ORM) for databases via Doctrine. The framework makes it relatively easy and secure to store a number of users and to provide login and registration forms.

4.2 Visualizations

In software engineering, it is good practice to keep content, design, and functionality separate. In web development, the fact that there are three different languages for the three different purposes on the client side encourages this pattern. Regarding the visualization of color spaces, Listing 4.1 provides an exemplary container within the content of a webpage for a figure with a visualization to appear in. The key element is the `div` tag in line 2 with the classes `visualization` and

`rgb-cube`. When the page is initialized, the Javascript responsible for functionality concerning visualizations scans the page for HTML elements with these two classes. Inside each such element a new RGB cube visualization can be initialized. The same holds for visualizations of any other color system. Additional elements indicate where controls like sliders or check boxes are to be placed. Not letting the visualization script place these containers automatically allows the content creator to decide whether or not to show these controls.

```
1 <div class="figure" id="fig-rgb-vis">
2     <div class="rgb-cube visualization aspect-ratio-preserver">
3         <img class="aspect-ratio" src="/resources/img/3by2aspect.
      png" />
4     </div>
5     <div class="figure-details">
6         <div class="selected-color"></div>
7         <div class="figure-title">
8             The RGB cube.<br/>
9             <span class="visualization-instructions">
10                Click and drag [...]
11            </span>
12        </div>
13        <div class="visualization-controls"></div>
14        <div class="visualization-controls-advanced"></div>
15    </div>
16 </div>
```

Listing 4.1: Example of HTML code for embedding the RGB cube visualization inside a webpage

Provided that all of this is working, it is only a small step to show two or more color systems side by side for comparison. Listing 4.2 shows the necessary change in Listing 4.1 for displaying both the RGB cube and the HSV color solid in the same figure.

```
1 <div class="aspect-ratio-preserver">
2     <img class="aspect-ratio" src="/resources/img/3by2aspect.png" /
      >
3     <div class="visualizations aspect-ratio-preservee">
4         <div class="rgb-cube visualization"></div>
5         <div class="hsv visualization"></div>
6     </div>
7 </div>
```

Listing 4.2: Replacement of lines 2 to 4 in Listing 4.1 for a comparison of color systems

As mentioned in Section 3.1, changes in one color system should be synchronized with the remaining color systems in a comparison. Figure 4.1 shows a structure of classes that enables this functionality. Each visualization holds a reference to

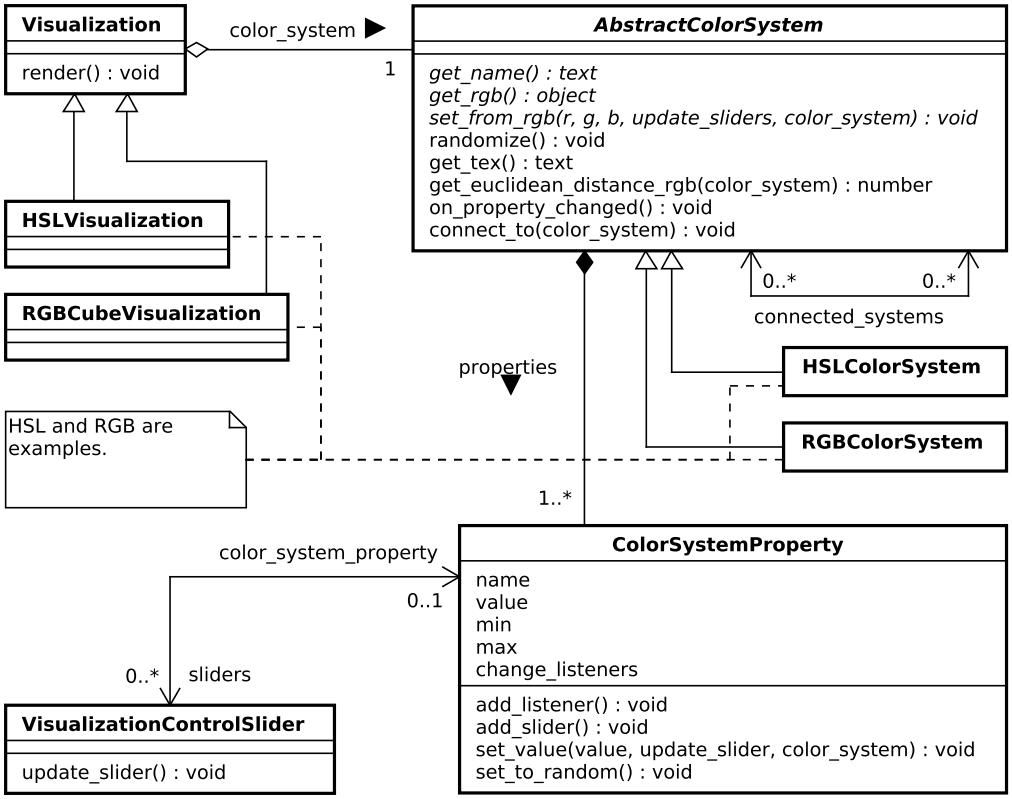


Figure 4.1: Class diagram for the composition of a Visualization with focus on the underlying representation of the color system

its own implementation of the `AbstractColorSystem` class which is responsible for converting colors from and to RGB. This will be crucial for the synchronization, since colors from almost all color spaces can be converted to and from RGB without much effort (see Figure 2.1). For this purpose, color systems can be connected to each other using the method `connect_to`. Each implementation of a color system consists of typically three or four numerical values which are stored in `ColorSystemProperty` objects, which in turn can be connected to a slider control. Therefore, whenever the user makes a change to one of the sliders, one of the color system properties will notify the color system which again notifies the visualization. Instead of drawing the visualization in an infinite loop like the scene of a game, it is sufficient to only update the screen when such an event occurs. Especially on mobile devices with limited battery capacity this has the benefit of lower power consumption.

The process of how a change in one slider control propagates to the visualizations and to the other sliders is shown in detail in Figure 4.2. As outlined above, when a user makes a change to a slider a chain of events is triggered. The slider controller sets the value of the corresponding color system property, which in turn notifies

the color system it belongs to. The color system class then performs two actions: First it notifies its parent visualization so that it may redraw itself. Then it uses the `set_from_rgb` method of all other color systems it is connected to in order to update their values. Along with this method call the first color system passes along a reference to itself which will be used to prevent an infinite loop of events. The other color systems can then set their properties and sliders to the new values as well as notify their respective visualizations.

4.3 Rendering

The simplest visualization as far as the implementation is concerned is that of the RGB cube. A solid as well as a wireframe cuboid are already provided in the Three.js library. As mentioned in Section 3.1, rather than drawing only a wireframe model as in Figure 2.3, it is possible to draw the opaque color solid. This has the benefit of not leaving the interpolation between only a few key colors to the user's imagination. Instead, they can see a representation of the color space as accurate as their display device allows.

One way to achieve this accurate visualization of the RGB color space is by using the programmable rendering pipeline. The idea is to use the shader programming language GLSL to write a fragment shader that determines the color of each pixel based on which portion of the color solid it is supposed to be showing. If the cube is placed in the world space as a unit cube between the coordinates $(0, 0, 0)^\mathsf{T}$ and $(1, 1, 1)^\mathsf{T}$, the fragment shader would simply need to interpret the current world coordinate as an RGB color. Since these coordinates are not provided by default, they have to be passed to the fragment shader by the vertex shader shown in Listing 4.3.

```
1 varying vec4 worldCoord;
2
3 void main() {
4     worldCoord = modelMatrix * vec4(position, 1.0);
5     gl_Position = projectionMatrix *
6                   modelViewMatrix *
7                   vec4(position, 1.0);
8 }
```

Listing 4.3: The vertex shader for all color systems. The matrices and the variable called position are provided by Three.js.

The only difference to the default vertex shader is in lines one and four. Since the variable `worldCoord` is declared as `varying` it will be automatically interpolated between vertices for the individual fragments, which means no additional calculations are required in the fragment shader in Listing 4.4. As a last step, these two shaders need to be applied to the cube as a Three.js shader material.

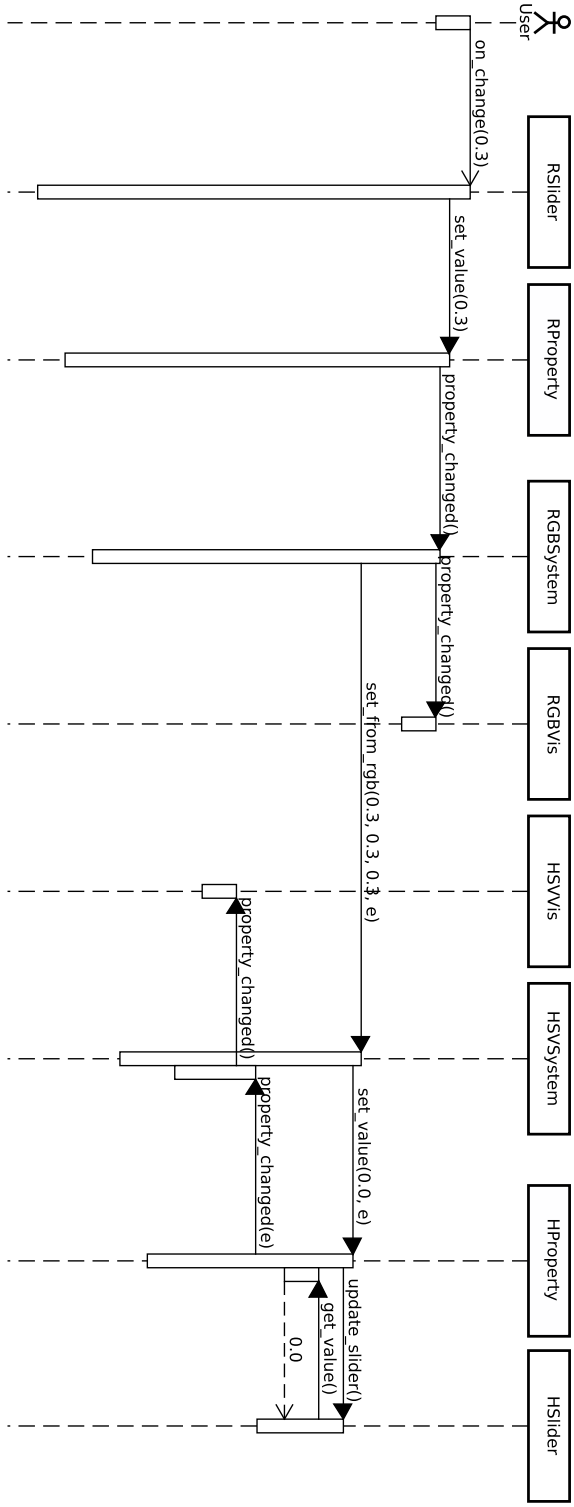


Figure 4.2: Sequence diagram showing an instance of how a change to the slider for red in an RGB visualization is propagated to a connected hue slider in an HSV visualization

```
varying vec4 worldCoord;

void main() {
    gl_FragColor = vec4(
        worldCoord.x,
        worldCoord.y,
        worldCoord.z,
        1.0
    );
}
```

Listing 4.4: The fragment shader for the RGB unit cube

Figure 4.3 shows a screenshot of this visualization after some additional elements have been added, which include labeled axes, a colored patch indicating the currently selected color as well as a bounding wireframe around the solid cube. The latter serves two functions. On the one hand, the wireframe clearly marks the edges of the cube which would otherwise be harder to see without any lighting and given the smooth transition of colors. On the other hand, it serves as an indicator of the maximum dimensions of the RGB cube when the cuboid inside is scaled to a different size. This scaling is done in correspondence to the currently selected color and it grants the user a look at the colors inside the color solid.

Visualizations of other color systems are rendered in a very similar way. In the case of HSL and HSV, one major difference is that world coordinates do not correspond to color coordinates as directly as they do for RGB and CMY, unless one chooses to interpret HSL and HSV as cubes as well. Considering here the example of HSV, the world coordinates within either the cylinder or the cone need to be translated into the hue angle, saturation, and value. In the fragment shader in Listing 4.5, this transformation is performed by the function `get_hsv`.

```
varying vec4 worldCoord;
uniform float radiusBottom;
uniform float radiusTop;
const float PI = 3.141592653589793;
const float Y_TRANSLATE = .5;

vec3 get_hsv() {
    float distFromY = distance(vec2(worldCoord.x, worldCoord.z),
                               vec2(0, 0));
    float v = worldCoord.y + Y_TRANSLATE;
    float s = distFromY / mix(radiusBottom, radiusTop, v);
    float h = (atan(worldCoord.z, -worldCoord.x) + PI)
              / (PI * 2.0);
    return vec3(h, s, v);
}

vec3 hsv_to_rgb(in vec3 hsv) { ... }

```

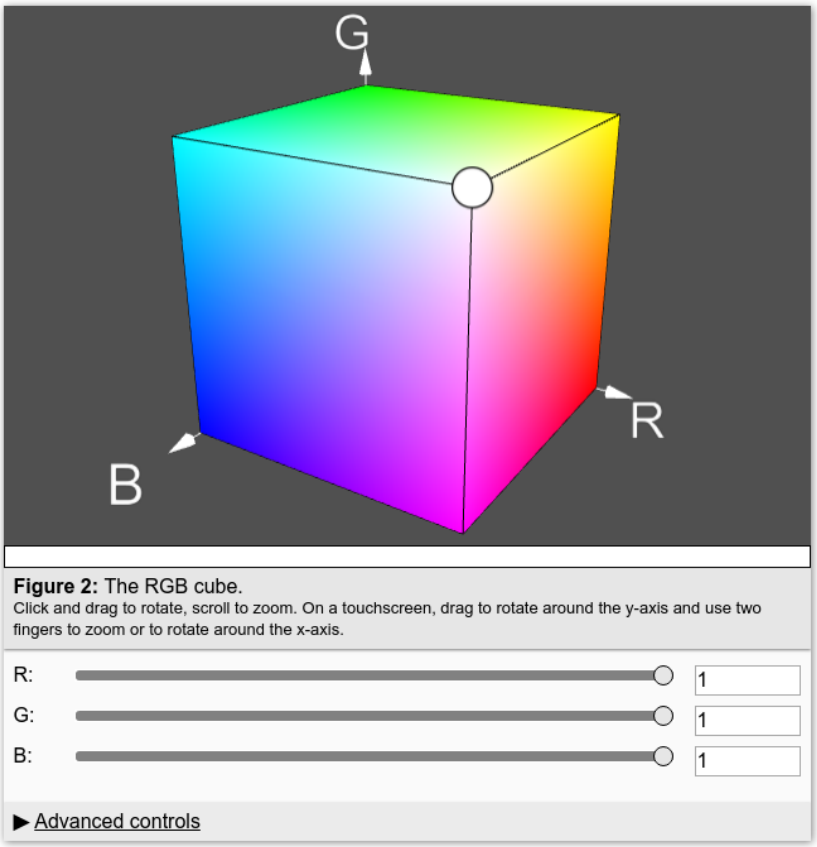


Figure 4.3: Screenshot of the implemented RGB cube visualization

```
19 void main() {
20     vec3 rgb = hsv_to_rgb(get_hsv());
21     gl_FragColor = vec4(rgb.x, rgb.y, rgb.z, 1.0);
22 }
```

Listing 4.5: The fragment shader for the HSV cylinder or cone.

The transformation assumes a cylinder-like object of unit height centered around the origin, its bases parallel to the OpenGL XZ plane. This makes it easy to compute the HSV value, which is simply the current Y coordinate offset by 0.5. The shader is given two radii for the bottom and the top of the cylinder-like object as input variables. This removes the need to write two separate shaders for the cylinder and the cone representation since the HSV cone is equivalent to a cylinder with bottom radius equal to zero. The saturation is therefore obtained by scaling the distance from the Y axis `distFromY` by the inverse of the current interpolation between the two radii. Finally, the hue can be calculated using the inverse tangent. For the sake of brevity, the function `hsv_to_rgb` has been omitted in Listing 4.5 since its mathematical definition has already been given in Section 2.3.5.

4.4 Exercises

An exercise contains a number of tasks as shown in Figure 4.4. With the mixed tasks exercise of Section 3.3.4 in mind, this implementation allows to keep only one general exercise class with tasks of different types depending on the current configuration. All tasks are kept in a list. The first task is started by the exercise on initialization via a call of the method `next_task` which calls `run` on the first task in the list. This task is then initialized according to its implementation and eventually the user will click on a button labeled “Next”. If the answer was correct or the maximum number of attempts `max_attemps` has been reached, this click will result in the method `on_task_finished` to be called in the parent exercise. The responsibility of this method then is to record the correctness of the just completed task for the final summary and to start the next task if one is available. Figure 4.4 shows classes for the following types of exercises discussed in Section 3.3:

Color matching exercise: If an instance of the class `Exercise` contains only tasks which are an instance of `ColorMatchingTask`, it would correspond to a color matching exercise as described in Section 3.3.1. The class `AbstractColorSystem` of Section 4.2 is re-used to handle the representation, randomization and conversion of the current color and the target color.

Color conversion exercise (advanced): In a broader sense, an advanced color conversion as described in Section 3.3.3 can also be interpreted as color matching with numerical rather than visual representations of the target and the current color. Therefore, the corresponding task is also implemented in the class `ColorMatchingTask`. In this case, the flag `is_conversion_task`

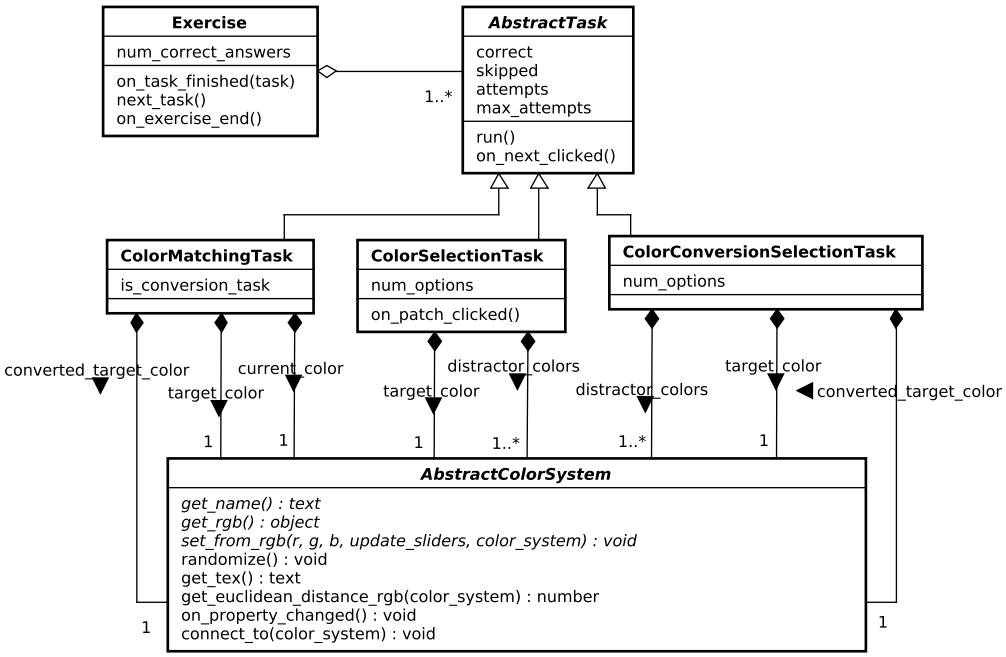


Figure 4.4: Class diagram for an exercise and its tasks

will be true and the `converted_target_color` will be used in addition to the other two color system references to display the correct numerical result if the user answered incorrectly too many times.

Color conversion exercise (medium): The task corresponding to this exercise type is implemented in the class `ColorConversionSelectionTask`. The attribute `num_options` determines how many options, including the target color, are to be shown.

Color selection exercise: This exercise type discussed in Section 3.3.2 is implemented in the class `ColorSelectionTask`.

The configuration of an exercise, like that of a visualization, can be adjusted in the HTML code of the surrounding webpage. Listing 4.6 demonstrates this in the example of a mixed tasks exercise with color selection and color matching tasks. For every `div` element in a page, a new instance of the JavaScript class `Exercise` is created. This class expects two HTML data attributes for said `div` element: One which indicates the total number of tasks after which this exercise is supposed to end, and another which lists the possible task types. The latter is represented as an array in JavaScript Object Notation (JSON).

```
<div class="exercise"
    data-num-rounds="15"
    data-task-types='[
        {
            "name": "ColorSelection",
            "options": {
                "color_systems":
                    ["rgb", "hsl", "hsv", "cmy", "cmyk"],
                "show_visualization": false,
                "max_attempts": 3,
                "allow_skip_after_first_attempt": true,
                "num_options": 8
            },
            "weight": 1
        },
        {
            "name": "ColorMatching",
            "options": {
                "color_systems":
                    ["rgb", "hsl", "hsv", "cmy", "cmyk"],
                "show_visualization": true,
                "max_attempts": 3,
                "allow_skip_after_first_attempt": true,
                "show_current_color": true,
                "show_target_color": true
            },
            "weight": 1
        }
    ]'
>
</div>
```

Listing 4.6: HTML and JSON code for a mixed exercise consisting of color matching and color selection tasks

5 Evaluation

In order to assess the quality of the developed prototype and, as a side-effect, to gather user feedback, an evaluation has been conducted. On the one hand, the goal was to quantify the usability of the system. On the other hand, the evaluation was to answer the question whether the visualizations and exercises have a positive effect on learning outcomes. This chapter describes the methodology of the evaluation. Furthermore, the results are presented and discussed.

5.1 Method

The evaluation was conducted in the form of a homework assignment for a lecture on computer graphics titled „Grundlagen der Computergrafik“. The assignment consisted of two tasks. In the first task, students were asked to use the website to acquaint themselves with color models and color spaces in general, as well as with RGB, CMY, CMYK, HSL, and HSV in particular. To fulfill this part of their homework, students were required to complete each of the exercises Color Matching, Selection, Conversion Selection, and Conversion at least once. At the time of evaluation, each of these exercises consisted of ten tasks. The second task was to fill out a Google Forms[1] questionnaire. Because class attendance is not mandatory and with the intention for all students to be roughly on the same level in the beginning, the topic of color theory was not discussed in detail in class before said homework assignment.

For evaluating the learning effect, students were randomly assigned to one of two groups:

Group A: This group would only be able to access visualizations and exercises concerned with RGB, CMY, and CMYK. On the pages about HSL and HSV they would see the same static images as are used in this document.

Group B: This group would only be able to access visualizations and exercises concerned with RGB, HSL, and HSV. There was no replacement for the CMY visualization.

Group A and group B were therefore each both control and experimental group, depending on the topic as shown in Table 5.1.

The assignment of each student to one of these groups and the subsequent management of access to certain content was realized by means of the students'

[1] `https://www.google.com/forms/about/`

Table 5.1: Experiment design

	RGB/CMY/CMYK	**RGB/HSL/HSV**
Group A	Experiment	Control
Group B	Control	Experiment

university email addresses. Participants were required to enter their university username in order to register for the homework assignment. They then automatically received a personalized key via email with which they could log on to the website.

For the assessment of usability, the questionnaire included the questions for the System Usability Scale (SUS) [Bro+96, pp. 4f.]. The SUS has been used in a large number of studies over the years [BKM08, p. 576] and is considered a valid and reliable tool for measuring usability, especially considering the relatively small number of questions [BKM08, pp. 581f., 588]. It consists of ten statements which the subject must respond to on a five-point Likert scale ranging from "strongly disagree" to "strongly agree" [Bro+96, p. 5]. In order to distinguish between the usability of the visualizations and that of the exercises, the questionnaire included one set of SUS questions for each.

For the purpose of testing the hypothesis that the visualizations and exercises do have a positive effect on learning outcomes, the questionnaire also included ten color conversion tasks. These tasks were similar to the tasks in the color conversion selection exercise in that for each question five options were given. In each case only one of the five options was correct. Of the ten tasks, five were to convert between RGB, CMY, or CMYK, while the other five dealt with conversions between RGB, HSL, and HSV. Both experimental groups received the same questionnaire, which is attached in Appendix A.

To discourage the students from cheating in the questionnaire, they were reminded that their fulfillment of the homework requirements would not be influenced by their performance in said tasks. In the weeks after the evaluation, the content restrictions for both experimental groups have been removed. This was done in order to give every student as equal as possible an opportunity to study in the event that color conversions will be part of the final exam.

5.1.1 Participants

In total, 56 students attending the computer graphics lecture mentioned above participated in the evaluation. Two of them were excluded from the analysis because they each completed the questionnaire twice and both times reported to have seen the questions for the first time. The lecture was given in German. Participants were between 19 and 29 years old (average $M = 22.1$, standard deviation $SD = 2.2$), eight of them being female. The distribution of participants across groups is summarized in Table 5.2. Three participants reported having a form of color vision deficiency. Since most of the product's contents except for the

color matching exercise do not rely on color vision alone, the three participants were not excluded. Of the 56 students, only 2 reported that they used a tablet computer as their primary device to access the website. Another student checked the option "other touch screen device".

Table 5.2: Demographical data of participants by group

	Group A	**Group B**	**Both groups**
Participants	27	27	54
# female	4	4	8
# male	23	23	46
Age: M	22.0	22.1	22.1
Age: SD	2.1	2.3	2.2
Color vision deficiencies	2	1	3

5.1.2 Statistical Analysis

SUS scores were calculated on a per-subject basis as described by Brooke et al. [Bro+96, p. 6] and the arithmetic means were calculated for the visualization ratings as well as for the exercise ratings. The worst possible SUS rating is 0 and the best possible rating is 100. However, SUS ratings should not be confused with the percentile rank; a score of 50, for example, does not indicate a system with better usability than half of all possible systems. It is also important to note that responses to the SUS questions should not be analyzed for each question individually [Bro+96, p. 6]. Instead, the aforementioned average ratings can be set in relation to other SUS scores based on an analysis of about 200 previous SUS studies [BKM08, p. 575, 592]. In most of these roughly 200 studies, the questions were modified to use the word "awkward" rather than "cumbersome" and the word "product" instead of "system" [BKM08, p. 576]. The majority of attendees of the lecture are not native English speakers and they were free to look up unknown words while answering the questionnaire. Therefore, it can be assumed that the comparison of SUS scores of this analysis with those obtained using modified questions is still valid.

In order to test the hypothesis of Section 5.1, Welch's one-sided t-test was conducted with independent samples. Specifically, the null hypothesis is the following:

> *Students will answer fewer or the same number of questions about a set of color spaces $\mathcal{C}$ correctly if they had access to $\mathcal{C}$'s visualizations and exercises.*

5.2 Results

On average, the visualizations received an SUS score of 74.5 ($SD = 13.1$) and the exercises received a score of 66.2 ($SD = 19.0$). As is also visible in Figure 5.1,

the minimum and maximum SUS score for the exercises is less than the respective minimum and maximum score for the visualizations.

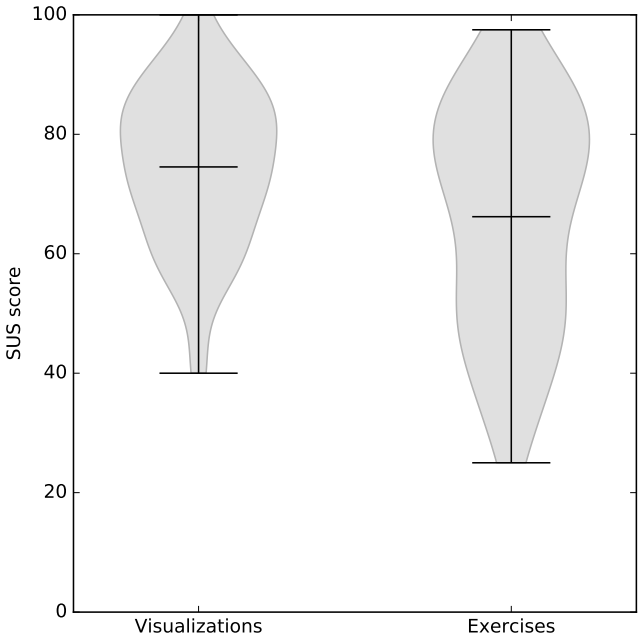


Figure 5.1: Collected SUS scores are visualized as violin plots. The horizontal bars indicate the minimum, the mean, and the maximum score.

In comparison to other SUS studies [BKM08, p. 592], the score for the visualizations falls into the third quartile, it is assigned an acceptability rating of "acceptable" and an adjective rating of "good". The SUS score for the exercises lies within the second quartile. Its acceptability is considered "marginal" (albeit on the "high" end), its adjective rating is "OK".

In the color conversion tasks of the questionnaire dealing with the color spaces RGB, CMY, and CMYK, students of group A (who received the visualizations and exercises for said color spaces) answered an average of 4.5 questions correctly ($SD = 1.1$), whereas students of group B did 3.6 ($SD = 1.8$). This difference is significant with t-test results of $t(44.3) = 2.2$ and $p = 0.018 < 0.05$.

On the other hand, in the RGB, HSL, and HSV conversions, group A responded correctly to 2.4 questions on average ($SD = 1.4$), while group B responded correctly to 2.6 questions ($SD = 1.8$). As expected, group B does answer more questions correctly than group A in this category, but the difference is not significant; $t(49.0) = 0.4, p = 0.337 \geq 0.05$. These results are reflected in Figure 5.2.

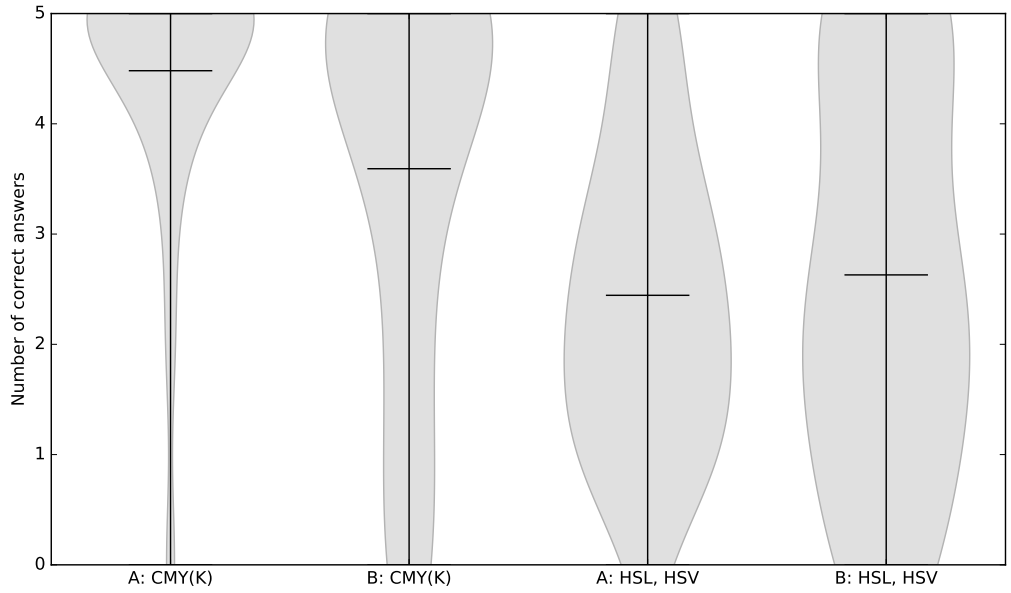


Figure 5.2: The numbers of correct answers by experimental group and condition are visualized as violin plots. Horizontal bars indicate the average.

5.3 Discussion

In absolute terms, the number of correctly answered conversion questions indicates that students did seem to have a good understanding of the color systems involved at the end of the evaluation. Had they simply guessed the answer to each conversion, at five options per question they would have been expected to only answer one conversion correctly per category (see Figure 5.2). Whether this understanding was present before they used the product can not be determined from the present data.

Students converted more colors correctly on average if they previously had access to interactive visualizations and exercises for the respective color systems involved in the conversion. This difference was significant in the case of questions about the RGB, CMY, and CMYK color spaces and therefore indicates a positive effect on learning. However, providing visualizations and exercises did not show a significant effect when it came to questions about RGB in combination with HSL and HSV. In the following, factors that might have influenced the evaluation will be discussed. Whether those factors did indeed have an effect on the outcome and in which way would have to be verified in further studies.

Because the evaluation was conducted among students in a class on computer graphics, the set of participants is very homogeneous in terms of age, gender, and level of education. The benefit of this is that the product could be tested on a group

of subjects closest to the primary target audience. On the other hand, it is now unclear how accessible the product will be to students who are not as familiar with computer science, or to a younger audience still attending school. Furthermore, due to the method of evaluation, the visualizations and exercises were most likely not used in the exact same way as they normally would have been: As discussed in Section 2.2, it would make sense to use the product as an augmentation of the lecture, which in this case was in part avoided for reasons outlined in Section 5.1. Furthermore, for students who wanted the full homework points, participation was mandatory. It was useful for the homework assignment and for the questionnaire as well to require the completion of four specific exercises. This ensured that every participant had used the various exercises not just superficially. However, an average user might not finish all tasks of every exercise because, for instance, they realize sooner that they already understand the concept. These facts might have been part of the reason why seven participants commented on the number of exercise tasks being too high. In general, it would also be interesting to see how the difference between groups develops after a longer period of time with students not being exposed to the topic in the meantime.

Another thing to keep in mind about the SUS scores is that no single participant had seen the product in its entirety when they submitted their rating. In addition, since the SUS is supposed to be a very general solution for evaluating different products, some of the ten statements are not clearly defined in the context of the product of this thesis. For example, one might assume that the statement "I think that I would need the support of a technical person to be able to use this system" would be mostly rejected by a group of people who can already be considered technical persons. A further example is the statement "I needed to learn a lot of things before I could get going with this system," which, depending on one's interpretation, one would assume to be answered mostly affirmatively, as learning is one of the goals of this product.

Nevertheless, this evaluation has provided useful insight. The usability acceptability rating of "acceptable" or better indicates that development of the product has been going in the right direction. Many participants also used the opportunity of the input field for additional feedback at the end of the questionnaire (see Appendix B) or wrote helpful emails with suggestions for improvement. Finally, it is noteworthy that there have been no complaints about major malfunctions in the software or about the visualizations and exercises not running at all on a given system.

6 Conclusion

The aim of this thesis was to develop a learning tool for students to study and practice certain aspects of color science with a focus on color models and color spaces (see Section 1.2). This aim was formulated in light of a previous solution not being easily accessible with modern web browsers anymore. The finished prototype appears to be functional and adequately usable on full-sized computers as well as on mobile devices, based on user feedback.

One challenging aspect of the implementation was getting acquainted with the languages and tools necessary for developing a hopefully robust web application (see Section 4.1). Especially the feature-rich Symfony framework can be challenging to learn with little previous knowledge of PHP. At the same time, however, one can easily appreciate the merit of said languages and tools. The Twig templating engine, for example, is very helpful for producing HTML code in that it makes the process of writing HTML much less repetitive and better organized. Similar benefits come with Less CSS compared to regular CSS, and with the method of transpiling Javascript and merging many files into one.

The main focus of the prototype lies on the systems RGB, CMY(K), HSL, and HSV. This is partly due to the syllabus of the computer graphics lecture and partly because some of the other systems would be more challenging to visualize or to incorporate in the generalized exercises. However, as has been outlined in the concept in Chapter 3, these challenges can certainly be overcome.

The visualizations themselves could be implemented very closely to their conceptualization in Chapter 3. Most notably, thanks to the GLSL shaders presented in Section 4.3, a look inside each implemented color space could be realized. The largest missing feature as of yet is the possibility for users to fully configure their own comparison of visualizations.

The exercises color matching, selection, and conversion have been mostly implemented as described in Section 3.3. Due to the generalized nature of these exercises, it should be easy to make them available for any color systems that may be implemented in the future. In the current state of the prototype, the exercises can be and have been adapted to different contexts in regards to difficulty, included tasks, and featured color systems. Furthermore, there are now exercise settings a user can indeed change, such as the number of options in color selection tasks or whether to display hints in color matching tasks. It is hoped that these flexibilities will allow students to somewhat adapt the difficulty and contents of exercises to their progress, thereby reducing frustration and maintaining motivation. Not yet implemented are the specialized mixed exercise tasks outlined in Section 3.3.4. These would be of most value once other color spaces like CIE XYZ,

CIELAB, Munsell, or YCbCr are implemented.

Bibliography

[06] *OpenGL 2.1 Reference Pages*. `https://www.opengl.org/sdk/docs/man2/xhtml/glColor.xml`. Accessed: 2017-01-05. 2006.

[10] *Wavelengths of Visible Light*. National Aeronautics and Space Administration, Science Mission Directorate, `http://missionscience.nasa.gov/ems/09_visiblelight.html`. Accessed: 2016-09-13. 2010.

[11a] *Recommendation ITU-R BT.601-7: Studio encoding parameters of digital television for standard 4:3 and wide-screen 16:9 aspect ratios*. 2011.

[11b] *Recommendation ITU-T T.871: Information technology – Digital compression and coding of continuous-tone still images: JPEG File Interchange Format (JFIF)*. 2011.

[13] *How to interpret the sRGB color space (specified in IEC 61966-2-1) for ICC profiles*. World Wide Web Consortium, `https://www.w3.org/Graphics/Color/srgb`. Accessed: 2016-08-24. 2013.

[15a] *Recommendation ITU-R BT.2020-2: Parameter values for ultra-high definition television systems for production and international programme exchange*. 2015.

[15b] *Recommendation ITU-R BT.709-6: Parameter values for the HDTV standards for production and international programme exchange*. 2015.

[16] *import - JavaScript*. Mozilla Developer Network, `https://developer.mozilla.org/en/docs/Web/JavaScript/Reference/Statements/import`. Accessed: 2016-10-22. 2016.

[AKM07] Ahirwal, Balkrishan, Khadtare, Mahesh, and Mehta, Rakesh. "FPGA based system for color space transformation RGB to YIQ and YCbCr." In: *International Conference on Intelligent and Advanced Systems 2007*. IEEE. 2007, pp. 1345–1349.

[BB09a] Burger, Wilhelm and Burge, Mark J. *Principles of Digital Image Processing: Fundamental Techniques*. 1st. Undergraduate Topics in Computer Science. London: Springer-Verlag London, 2009. ISBN: 978-1-84800-190-9, 978-1-84800-191-6.

[BB09b] Burger, Wilhelm and Burge, Mark J. *Principles of Digital Image Processing: Core Algorithms*. 1st. Undergraduate Topics in Computer Science. London: Springer-Verlag London, 2009. ISBN: 978-1-84800-194-7, 978-1-84800-195-4.

[BKM08] Bangor, Aaron, Kortum, Philip T, and Miller, James T. "An empirical evaluation of the system usability scale." In: *Intl. Journal of Human–Computer Interaction* 24.6 (2008), pp. 574–594.

[Bro+96] Brooke, John et al. "SUS-A quick and dirty usability scale." In: *Usability evaluation in industry* 189.194 (1996), pp. 4–7.

[Ç+11] Çelik, Tantek, Lilley, Chris, Baron, L David, et al. *CSS Color Module Level 3: W3C Recommendation 07 June 2011.* World Wide Web Consortium, `http://www.w3.org/TR/2011/REC-css3-color-20110607`. Accessed: 2016-08-24. 2011.

[DBM83] Dartnall, Herbert JA, Bowmaker, James K, and Mollon, John D. "Human visual pigments: microspectrophotometric results from the eyes of seven persons." In: *Proceedings of the Royal Society of London B: Biological Sciences* 220.1218 (1983), pp. 115–130.

[GRR10] Ganesan, P, Rajini, V, and Rajkumar, R Immanuvel. "Segmentation and edge detection of color images using CIELAB color space and edge detectors." In: *Emerging Trends in Robotics and Communication Technologies (INTERACT), 2010 International Conference on.* IEEE. 2010, pp. 393–397.

[Gui32] Guild, John. "The colorimetric properties of the spectrum." In: *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character* 230 (1932), pp. 149–187.

[HSF06] Hrehorova, Erika, Sharma, Abhay, and Fleming, PD. "Color reproduction studies in RGB and CMYK workflows using inkjet printer drivers and RIPs." In: *Proc. 58th TAGA Annual Technical Conference, Vancouver, British Columbia.* Citeseer. 2006, pp. 159–171.

[JG78] Joblove, George H and Greenberg, Donald. "Color spaces for computer graphics." In: *ACM siggraph computer graphics.* Vol. 12. 3. ACM. 1978, pp. 20–25.

[Jud+64] Judd, Deane B et al. "Spectral distribution of typical daylight as a function of correlated color temperature." In: *Josa* 54.8 (1964), pp. 1031–1040.

[Man+16] Mansencal, Thomas et al. *Colour 0.3.8.* `http://dx.doi.org/10.5281/zenodo.57294`. July 2016. doi: `10.5281/zenodo.57294`. url: `http://dx.doi.org/10.5281/zenodo.57294`.

[May+95] Mayer, Richard E et al. "A generative theory of textbook design: Using annotated illustrations to foster meaningful learning of science text." In: *Educational Technology Research and Development* 43.1 (1995), pp. 31–41.

[May02] Mayer, Richard E. "Multimedia learning." In: *Psychology of learning and motivation* 41 (2002), pp. 85–139.

[McL76] McLaren, K. "XIII—The Development of the CIE 1976 (L* a* b*) Uniform Colour Space and Colour-difference Formula." In: *Journal of the Society of Dyers and Colourists* 92.9 (1976), pp. 338–341. ISSN: 1478-4408. DOI: 10.1111/j.1478-4408.1976.tb03301.x. URL: http://dx.doi.org/10.1111/j.1478-4408.1976.tb03301.x.

[Mea+09] Means, Barbara et al. "Evaluation of evidence-based practices in online learning: A meta-analysis and review of online learning studies." In: *US Department of Education* (2009).

[MM99] Moreno, Roxana and Mayer, Richard E. "Cognitive principles of multimedia learning: The role of modality and contiguity." In: *Journal of educational psychology* 91.2 (1999), p. 358.

[Mor03] Moroney, Nathan. "A hypothesis regarding the poor blue constancy of CIELAB." In: *Color Research & Application* 28.5 (2003), pp. 371–378.

[Mun12] Munsell, Albert H. "A pigment color system and notation." In: *The American Journal of Psychology* 23.2 (1912), pp. 236–244.

[NNJ43] Newhall, Sidney M, Nickerson, Dorothy, and Judd, Deane B. "Final report of the OSA subcommittee on the spacing of the Munsell colors." In: *JOSA* 33.7 (1943), pp. 385–418.

[Ope16] OpenGL Wiki contributors. *Legacy OpenGL — OpenGL Wiki*. http://www.khronos.org/opengl/wiki_opengl/index.php?title=Legacy_OpenGL&oldid=13721. Accessed: 2017-01-05. 2016.

[Pou16] Poushter, Jacob. "Smartphone Ownership and Internet Usage Continues to Climb in Emerging Economies." In: *Pew Research Center: Global Attitudes & Trends* (2016).

[SD03] Schröder, Monika and Domik, Gitta. „Veränderungen von Lehreinheiten durch veränderte Ansprüche am Beispiel ‚Computerbilder'." In: *DeLFI 2003: Die 1. e-Learning Fachtagung Informatik*. Ed. by Bode, Arndt et al. Bonn: Gesellschaft für Informatik, 2003, pp. 402–411.

[SG31] Smith, Thomas and Guild, John. "The CIE colorimetric standards and their use." In: *Transactions of the Optical Society* 33.3 (1931), p. 73.

[Tig08] Tigges, Anja. *Geschlecht und digitale Medien. Entwicklung und Nutzung digitaler Medien im hochschulischen Lehr-/Lernkontext*. 1st. VS Verlag für Sozialwissenschaften, 2008. Chap. Studie 3: Entwicklung und Nutzung des Lernmoduls „Computergenerierte Farbe" des Projekts SIMBA, pp. 163–186. ISBN: 978-3-531-15707-8. DOI: 10.1007/978-3-531-90812-0.

[Wir15] Wirfs-Brock, Allen. *ECMAScript 2015 Language Specification*. ECMA International, `http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-262.pdf`. Accessed: 2016-10-22. 2015.

[WMN12] Woodcock, Ben, Middleton, Andrew, and Nortcliffe, Anne. "Considering the Smartphone Learner: an investigation into student interest in the use of personal technology to enhance their learning." In: *Student Engagement and Experience Journal* 1.1 (2012), pp. 1–15.

[Wri29] Wright, William David. "A re-determination of the trichromatic coefficients of the spectral colours." In: *Transactions of the Optical Society* 30.4 (1929), p. 141.

Appendices

A Questionnaire

These are all the items of the Google Forms questionnaire mentioned in Section 5.1. They include the ten statements for the SUS, as well as the color conversion questions. Items not marked with an asterisk were optional.

Page "General information"

- Survey key

  IMPORTANT: You can view your survey key after logging in at color.lukas-stratmann.com (this is not your login key!). It will be used to ensure you fulfilled all criteria for the GdC exercise, so make sure you enter the correct one; your data entered in these forms will be treated anonymously.

- Age*
- Sex*
  - Female
  - Male
  - Other
- Do you have any form of color vision deficiency?*

  (As compared to the majority of humans. Please don't choose "Yes" just because you can't see IR or UV.)
  - No
  - Yes
- Field of study
  - Computer Science (Bachelor)
  - Computer Science (Master)
  - Other...
- What device did you primarily use to access the website?*
  - Laptop or desktop computer with keyboard and mouse
  - Smartphone
  - Tablet

  - Other touch screen device
  - Other...

Page “Color conversion”

It is important for my thesis that you answer these questions honestly and without cheating! If you are not sure about an answer, have a guess. This data will be treated anonymously.

- What is (0.5, 1.0, 0.5)_HSL in RGB?*
  - (1, 0, 0)_RGB
  - (0, 1, 1)_RGB
  - (1, 0, 1)_RGB
  - (0, 0.5, 0.5)_RGB
  - (0.5, 1, 1)_RGB
- What is (127, 255, 127)_RGB in HSL?*
  - (1/3, 1, 3/4)_HSL
  - (1/3, 1/2, 1)_HSL
  - (1/3, 3/4, 1)_HSL
  - (2/3, 1, 3/4)_HSL
  - (0, 3/4, 3/4)_HSL
- What is (0.2, 0.5, 0.8)_RGB in CMYK?*
  - (0.6, 0.3, 0.0, 0.2)_CMYK
  - (0.6, 0.3, 0.0, 0.8)_CMYK
  - (0.8, 0.5, 0.2, 0.2)_CMYK
  - (0.8, 0.5, 0.2, 0.8)_CMYK
  - (1.0, 0.7, 0.4, 0.2)_CMYK
- What is (1, 0, 1)_RGB in HSV?*
  - (4/6, 1, 1)_HSV
  - (1/6, 1, 1)_HSV
  - (5/6, 1, 1)_HSV
  - (1/6, 1, 1/2)_HSV
  - (5/6, 1, 1/2)_HSV
- What is (0.2, 0.4, 0.6)_CMY in CMYK?*

  - (0.2, 0.4, 0.6, 0.2)_CMYK
  - (0.0, 0.2, 0.4, 0.2)_CMYK
  - (0.2, 0.4, 0.6, 0.6)_CMYK
  - (0.0, 0.4, 0.6, 0.2)_CMYK
  - (0.2, 0.4, 0.0, 0.6)_CMYK
- What is (240°, 1, 0.5)_HSV in HSL?*
  - (120°, 0.5, 0.25)_HSL
  - (240°, 1, 0.5)_HSL
  - (240°, 1, 0.25)_HSL
  - (240°, 0.5, 0.5)_HSL
  - (120°, 1, 0.25)_HSL
- What is (0.3, 0.5, 0.2)_CMY in RGB?*
  - (0.5, 0.8, 0.7)_RGB
  - (0.8, 0.5, 0.7)_RGB
  - (0.2, 0.0, 0.3)_RGB
  - (0.7, 0.5, 0.8)_RGB
  - (0.9, 0.7, 1.0)_RGB
- What is (15, 0, 127)_RGB in CMY?*
  - (127, 0, 112)_CMY
  - (255, 127, 241)_CMY
  - (240, 255, 128)_CMY
  - (241, 255, 127)_CMY
  - (112, 127, 0)_CMY
- What is (0.7, 0.9, 0.0, 0.1)_CMYK in CMY?*
  - (0.8, 1.0, 0.9)_CMY
  - (0.8, 1.0, 0.1)_CMY
  - (0.6, 0.8, 0.0)_CMY
  - (0.4, 0.2, 0.1)_CMY
  - (0.2, 0.0, 0.9)_CMY
- What is (120°, 255, 63)_HSV in RGB?*
  - (255, 255, 255)_RGB
  - (0, 0, 63)_RGB

- (127, 63, 127)_RGB
- (63, 63, 0)_RGB
- (0, 63, 0)_RGB

Page “Usability: Visualizations”

This part is only about the three-dimensional visualizations, not the exercises (see next part)!

[Each of the following statements (excluding the feedback item) had five options ranging from “strongly agree” to “strongly disagree”.]

- (Visualizations:) I think that I would like to use this system frequently.*
- (Visualizations:) I found the system unnecessarily complex.*
- (Visualizations:) I thought the system was easy to use.*
- (Visualizations:) I think that I would need the support of a technical person to be able to use this system.*
- (Visualizations:) I found the various functions in this system were well integrated.*
- (Visualizations:) I thought there was too much inconsistency in this system.*
- (Visualizations:) I would imagine that most people would learn to use this system very quickly.*
- (Visualizations:) I found the system very cumbersome to use.*
- (Visualizations:) I felt very confident using the system.*
- (Visualizations:) I needed to learn a lot of things before I could get going with this system.*
- Feedback about the visualizations

Page “Usability: Exercises”

This part is only about the exercises!

[The items on this page are the same as those on the previous page with the word “visualizations” replaced by “exercises”.]

Page “Finished”

- Can the data you provided be used anonymously in the evaluation?*

  (This will not affect your homework scores, so please be honest!)

  - Yes, I followed all the instructions and I answered all questions honestly. This is the first time I filled out or saw this questionnaire.
  - No.

- Additional feedback

- Browser

  (In case you encountered any bugs. Remember that you can also send me an email in that case, which would make it easier for me to ask follow-up questions to try and fix the bug.)

B User Feedback

Listed here is the feedback collected in the questionnaire, organized by experimental group. Not listed are emails that were sent privately, of which there only were a few anyway.

B.1 Visualizations

Group A:

- Sehr schöne Darstellungen! Eine Colorpick-Funktion wäre genial
- Also mir persönlich nutzt diese 3D-Darstellung gefühlt so gut wie gar nichts... Ich stelle mir das anders vor, wüsste jetzt aber auch nicht, wie ichs besser visualisieren sollte.
- +

Group B:

- Ich bin begeistert
- For HSV and HSL you have to know that you can rotate the cone/"diamond" by drag and drop to see the top. Maybe if it was pre-tilted a bit you could see all dimensions of it.
- Sorry I haven't used them a lot because I was already familiar with all of the explained color models/spaces.
- Nice tool, but hard conversions exercises.

B.2 Exercises

Group A:

- Ich weiß nicht ob für den Lerneffekt wirklich 15 aufgaben pro berreich notwendig sind. Ich bin mir irgendwann einfach nurnoch dumm vorgekommen zum 1000. mal das selbe zu tun. Für mich hat das vielleicht einen Lerneffekt für das Kopfrechnen gehabt ansonsten, naja...
- Waren echt gut gemacht.
- Matching the right color was really difficult because the allowed difference is in my opinion very small.

- 10 Fragen fande ich persönlich zu viele, da die Aufgaben immer sehr gelich waren, wodurch ich nach 7 oder 8 Fragen oft dachte „Nicht noch mal die Regeler verschieben bis es ungefähr passt“
- Number of exercises was very high for doing the same thing over and over again
- Color Conversion Selection was too easy because only the first number was relevant most of the time. Color Conversions from CMY to CMYK is copying the exact same thing from CMY to CMYK if K=0 is allowed -> Predetermined K value would make it more interesting.
- Farbtolleranz bei Optischen Farbabgleich zu gering, sonst sehr schön

Group B:

- Es waren im Gesamten doch zu viele Aufgaben. 5 Fragen pro Bereich hätten auch gereicht, da das Prinzip ja immer gleich war.
- At the exercises, there could have been a “hint” box, where the formulas were written for example.
- Too many tasks with converting hsv to hsl which takes forever
- A lot of multiple choice questions are easy to bypass by only knowing a few rules of thumb: max{r,g,b} = v or H=H
- Worse grinding than vanilla WoW.
- The exercises were a good training for conversion for me. I think there were too many conversions to do. It took a lot of time to calculate, but the concept was already clear to me.
- Hard conversion exercises.

B.3 Additional Feedback

Group A:

- Wenn man während der Execises schon die ganze Zeit die selben Aufgabenarten rechnet und die selben dann während des Fragebogens nochmal rechnen soll finde ich das höchst fragwürdig und nervig.
- Keine Bugs, die das System beeinträchtigt hätten.
- Bei der Aufgabe , bei welcher man die passende Farbe zu der gegebenen angeben soll, war es schwer die richtige zu finden aufgrund des PC Monitors. In den Aufgaben gab es keine zu HSV/HSL. Somit bin ich mir nicht sicher ob ich das in diesem Fragebogen richtig ausgefüllt habe. Eine Übung dazu wäre hilfreich.

- Weniger Rechenaufgaben bitte ! Ich hatte es auch nach dem ersten mal verstanden. Ansonsten hat es mir geholfen, die Systeme und ihre Umrechnung zu verstehen.

Group B:

- Tolles Konzept und tolle Seite mit super Visualisierungen. Der Zeitaufwand für die Aufgaben war jedoch etwas zu hoch. Siehe Feedback von Seite 4.
- The precision of the numbers is too exact or maybe too abstract (like 0,999999999999999), maybe use percentages from 0 to 100 with one decimal place.
- I was already familiar with color spaces and color models. I really learned how to convert these in detail and with pen and paper :D
- HSV to HSL conversions take too much time
- Konversionen zwischen Intervall [0,1] und [0,255] auch [0, 2pi] sind nicht klar im Text!
- I did not find any bug. Nice work!
- Finde das System im großen und ganzen sehr gut aber eine Formelsammlung wäre noch gut, damit man nicht alles abschreiben muss.

C Changelog Since Evaluation

The following changes have been made to the prototype since the evaluation (in chronological order from earliest to most recent):

- The necessity to log in has been removed.
- As requested, the default unit for hues in the HSL and HSV color models is now degrees.
- The user has the option to choose their preferred units in visualizations.
- It has been pointed out in an email that sliders for visualizations accepted numbers outside the valid range. Another student who participated in the evaluation helped discover an issue in color matching tasks when CMYK was involved: Because the task would allow invalid CMYK colors, the calculation of distances between colors would often not match what the user saw on the screen. Sliders and number inputs now ensure that only valid numbers can be entered.
- As one of the subjects pointed out in the questionnaire (see Section B.3), the numeric precision could be very high in color representations. Numbers are now rounded according to the currently selected units.
- An options menu has been added to the beginning of every exercise (can be disabled in HTML). The number of tasks per exercise is now user-selectable (minimum: 3, default: 10).
- Optionally, the units for color representations can be randomly selected in each exercise task. This should make some tasks more challenging where previously solutions could be very obvious, e.g. the conversion from CMYK to CMY.
- As requested, hints will now be shown in color matching tasks when the user approaches the correct color. Hints are enabled by default but can be turned off in the options menu at the beginning of an exercise.
- Visualizations can be viewed in fullscreen.
- Feedback in color matching tasks is now displayed as a highlighted item among a short list of possible feedbacks. This is to make it more clear how accurate the current input is.
- As suggested in Section B.1, the initial viewing angle in visualizations has been changed.

- A presentation mode for exercises has been added. This allows a presenter to prepare a non-random reusable sequence of (color matching) tasks.