Universität Stuttgart
Fakultät Informatik

Visualisierung und
Interaktive Systeme

A Depth-Cueing Scheme Based on Linear Transformations in Tristimulus Space

Daniel Weiskopf und Thomas Ertl

Bericht Nr. 2002/08
September 2002

Universität Stuttgart
Fakultät Informatik
Abteilung Visualisierung und Interaktive Systeme
Breitwiesenstr. 20–22
D-70565 Stuttgart

email: weiskopf@informatik.uni-stuttgart.de
URL: http://wwwvis.informatik.uni-stuttgart.de/˜weiskopf

A Depth-Cueing Scheme Based on Linear Transformations in Tristimulus Space

Daniel Weiskopf* Thomas Ertl†

Visualization and Interactive Systems Group‡
University of Stuttgart

Abstract

We propose a generic and flexible depth-cueing scheme which subsumes many well-known and new color-based depth-cueing approaches. In particular, it includes standard intensity depth-cueing and rather neglected pure saturation depth-cueing. A couple of new combinations and variations of depth cues are presented. Their usefulness is demonstrated in many different fields of application, reaching from non-photorealistic rendering to information visualization. In addition to cues based on a geometric concept of depth, an abstract visualization approach in the form of semantic depth-cueing is proposed. Our depth-cueing scheme is based on linear transformations in the 3D tristimulus space of colors and on weighted sums of colors. Since all of the required operations are supported by contemporary consumer graphics hardware, the depth-cueing scheme can be implemented without performance cutbacks. Therefore, any real-time rendering application can be enriched by sophisticated depth-cueing.

CR Categories: I.3.7 [Three-Dimensional Graphics and Realism]: Color, shading, shadowing, and texture

Keywords: depth-cueing, visual perception, color, saturation, non-photorealistic rendering, visualization

1 Introduction

The retina of the human eye is only two-dimensional and therefore depth cues are essential for the visual perception of spatial structures. So-called primary and secondary cues can be distinguished. Primary cues include binocular disparity (stereoscopic viewing), convergence, and accommodation. On a computer system, these can only be implemented in the form of specialized hardware, such as shutter glasses, and are not widely available. Secondary cues include perspective, relative size, occlusion, shadows, texture, color gradients, motion parallax, and active movement. A description of these depth cues can be found, for example, in [17]. These cues do not rely on a specific hardware and can thus be used to improve depth perception in computer-generated images.

In this paper, we present a color-based depth-cueing scheme which is especially appropriate for non-photorealistic rendering techniques. In this context, the term non-photorealistic rendering is used in a general sense: It comprises various rendering methods which are determined by non-physics-based lighting models, such as technical illustrations, artistic rendering, information visualization, terrain visualization, or volume illustrations. Intensity depth-cueing is a well-known example of a color-oriented technique for enhancing depth perception. This approach creates an effect similar to viewing the scene through haze by dimming the colors of far-away objects [9]. Intensity depth-cueing is influenced by the aerial perspective, which has been used by painters for many centuries. Early examples are the work by the Flemish painter Jan van Eyck, e.g., [31]. In many drawings, an additional slight color modulation is noticable—remote objects experience a subtle blue shift [8]. These depth-cueing techniques can be easily implemented by exploiting hardware-accelerated fogging on contemporary graphics chips.

A change in saturation is another effect prominent in aerial perspective—the colors of far-away objects are less saturated. Nevertheless, this effect has been widely neglected in computer-generated imagery, although investigations of the human visual system [28] indicate that color gradients at isoluminance affect perceived depth and that a gradient in saturation is particularly effective. Therefore, changing saturation can be used as a means of depth-cueing.

In this paper, we introduce a generic depth-cueing scheme which comprises all of the above color-based cues. Despite of its generality, our approach is computationally simple and can be directly mapped to contemporary consumer graphics hardware. It is transparent to the implementation of 3D viewers and can readily be combined with other non-photorealistic rendering or visualization techniques, assisting the idea that graphical excellence is nearly always multivariate (Tufte [29]). In particular, we would like to promote the use of saturation gradients as a monocular depth cue. Unlike intensity-based cueing, saturation depth-cueing leaves intensity and contrast invariant and therefore preserves features in the scene. This is close to Tufte's strategy of the smallest effective difference, making visual distinctions as subtle as possible, but still clear and effective. Because of the flexibility of our scheme even more sophisticated applications become possible, some of which are presented in this paper. In particular, we show how depth perception is improved by combining different depth cues. Furthermore, we introduce *semantic depth-cueing* as an abstract formulation of depth-cueing.

The paper is organized as follows. Section 2 briefly reviews previous and related work. In Section 3, the basis of our depth-cueing approach is developed. The subsequent section contains typical choices of parameters for the generic scheme. In Section 5, the usefulness of depth-cueing is demonstrated in many different fields of application, reaching from non-photorealistic rendering to information visualization. The mapping of our scheme to graphics hardware and an example implementation is described in Section 6. The paper ends with a short conclusion. In the appendix, an explicit formulation of the color transformation required for saturation depth-cueing is included.

2 Previous and Related Work

In recent years, a lot of research has been conducted on perceptional issues in computer graphics. Interrante et al. [16], Healey et al. [14], and Tumblin and Ferwerda [30] give a comprehensive presentation of human visual perception and corresponding applications in computer graphics. Closely related to this paper is a field of

*weiskopf@informatik.uni-stuttgart.de

†ertl@informatik.uni-stuttgart.de

‡Visualization and Interactive Systems Group, University of Stuttgart, Breitwiesenstr. 20–22, D-70565 Stuttgart, Germany

research which deals with physiological, psychological, and usability aspects of depth perception. Various depth cues are described, for example, by Keley [17]. Depth cues are evaluated with respect to depth-related tasks: For example, Zhai et al. [36] investigate the partial-occlusion effect produced by semitransparent surfaces; Hubona et al. [15] evaluate effects of shadows on depth perception; Mereu and Kazman [20] even use audio feedback to improve a user's sense of depth perception.

Troscianko et al. [28] investigate the ability of subjects to judge depth from various color gradients. Their study reveals that both saturation and luminance gradients, which occur in natural scenes, allow depth perception. On the contrary, for example, a red–green hue gradient, which does not occur in nature, does not affect depth perception. Troscianko et al.'s work is the psychological motivation for saturation depth-cueing, which is one of the core applications of our depth-cueing scheme.

Another field of research related to this paper comprises aspects of non-photorealistic rendering. In recent years, a lot of research has been conducted on this topic. Gooch and Gooch [11] and Green et al. [13] give a comprehensive presentation of the state-of-the-art in non-photorealistic rendering. A large body of research is focused on pen-and-ink illustrations [25, 33, 19, 22, 23, 6], which can be used to combine colored surface rendering and accentuated edge drawings. Depth perception in these applications can be further enhanced with our depth-cueing scheme. Other non-photorealistic rendering techniques make explicit use of color tone variations and are perfectly suited for color-based depth-cueing. For example, Curtis et al. [4] imitate the appearance of watercolored images for non-photorealistic rendering. Gooch and co-workers[10, 12] facilitate airbrush techniques for automatic technical illustrations. Ebert and Rheingans [8] adapt non-photorealistic rendering techniques to volume illustrations and introduce a variation of intensity depth-cueing by adding a shade of blue to distant objects.

In a work on scientific visualization, Bailey and Clark [1] employ texture mapping in order to produce depth-dependent color transformations for stereo vision based on *ChromaDepth* glasses. Finally, the book by Ebert et al. [7] contains a description of atmospheric scattering that is a procedural texturing approach to aerial perspective.

3 Linear Transformations and Weighted Sum of Colors

In this section, the basic idea of our generic depth-cueing scheme is presented. Typical applications are derived in the subsequent sections. The development of this scheme is guided by the well-known intensity depth-cueing [9]. This standard model produces a final color from a weighted sum of an original input color and a user-specified background color.

In our approach, this model is extended by two new, crucial features. First, the weights for the combination of the two colors can be chosen independently for each of the three color components. Secondly, the weighted sum does not need to be calculated with respect to a fixed color coordinate system (which would usually be RGB). In fact, an arbitrary linear transformation of the input colors is applied before the summation is performed.

The depth-cueing scheme of this paper is based on some fundamental concepts of color vision and colorimetry. Here, only some important relevant aspects are briefly reviewed. More detailed background information on color science can be found, for example, in the book by Wyszecki and Stiles [35].

A fundamental result of experimental color matching and a key element of colorimetry is the the so-called trichromatic generalization. Trichromatic generalization states that, over a wide range of observations, many color stimuli can be matched by an additive mixture of three fixed primary stimuli. All other stimuli have to be mixed with one or two primary stimuli before a complete match with the remaining stimuli can be achieved. In the stronger form of trichromatic generalization, linearity laws of additivity and proportionality are valid, giving rise to a three-dimensional vector space—the so-called tristimulus space of colors. From a physiological point of view, this property of color perception is based on the fact that three different types of receptors—three kinds of cones—are responsible for color detection in the human eye's retina. Neglecting higher levels of processing in the human visual system, signals from these receptors essentially add up to the finally perceived color.

A couple of different color spaces are in wide use today. One of these was introduced by the CIE (Commission International de l'Eclairage) for measurements of their CIE 1931 Standard Colorimetric Observer. It consists of the primaries red, green, and blue as monochromatic stimuli of wavelengths $\lambda_{\text{red}} = 700$ nm, $\lambda_{\text{green}} = 546.1$ nm, and $\lambda_{\text{blue}} = 435.8$ nm. Since the phosphors of contemporary computer monitors are not identical to these CIE RGB primaries, related RGB systems may be based on measurements of these phosphors. Another important color system is the XYZ system, which was introduced as a standard colorimetry system by the CIE.

All these color systems are directly based on the vector space concept of tristimulus space. They just represent different basis vectors—the primary colors—of the same three-dimensional vector space. Therefore, a change of basis can be realized by a linear transformation. Suppose that an arbitrary color $\mathbf{C}$ is given with respect to a basis of primary colors, $\{\mathbf{P}_i \,|\, i = 1,2,3\}$, and can accordingly be represented as a three-component tristimulus vector (C_1, C_2, C_3):

$$\mathbf{C} = \sum_{i=1}^{3} C_i \mathbf{P}_i \quad .$$

Let us assume that the old basis $\{\mathbf{P}_i \,|\, i = 1,2,3\}$ and a new basis $\{\tilde{\mathbf{P}}_i \,|\, i = 1,2,3\}$ are related by

$$\mathbf{P}_j = \sum_{i=1}^{3} M_{ij} \tilde{\mathbf{P}}_i \quad ,$$

where M_{ij} are the elements of a 3×3 matrix. Then, the color $(\tilde{C}_1, \tilde{C}_2, \tilde{C}_3)$ represented with respect to the new basis $\tilde{\mathbf{P}}_i$ can be expressed in terms of the transformation matrix M and the original tristimulus vector:

$$\tilde{C}_i = \sum_{j=1}^{3} M_{ij} C_j \quad ,$$

i.e., the linear transformation between different representations is accomplished by the matrix M.

In this notation, our depth-cueing scheme can be formulated as follows. As input data, an original color $\mathbf{C}_{\text{orig}}$ and an admixing color $\mathbf{C}_{\text{mix}}$ shall be given with respect to an arbitrary, but fixed basis $\{\mathbf{P}_i\}$. As contemporary graphics hardware and most modeling and design software is based on RGB, the basis $\{\mathbf{P}_i\}$ can usually be identified with RGB. Furthermore, a second basis $\{\tilde{\mathbf{P}}_i\}$ or the transformation matrix M have to be supplied. Appropriate choices for the second basis are discussed in the following section.

Figure 1 shows the structure of the depth-cueing scheme. The green part of the process has to be performed only once in a preprocessing step, whereas the blue part has to be evaluated for each single pixel in the image plane. In the first step of the process, the original color $\mathbf{C}_{\text{orig}}$ and the admixing color $\mathbf{C}_{\text{mix}}$ are transformed

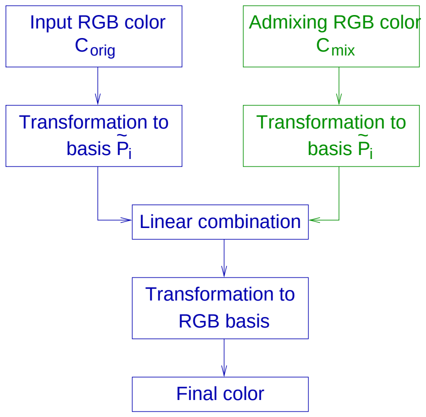
Input RGB color C_{orig}
Admixing RGB color C_{mix}
Transformation to basis $\tilde{P}_i$
Transformation to basis $\tilde{P}_i$
Linear combination
Transformation to RGB basis
Final color

Figure 1: Structure of the depth-cueing scheme. The green part is evaluated in a preprocessing step, the blue part is processed for each single pixel.

into the second basis,

$$\tilde{C}_{\text{orig},i} = \sum_{j=1}^{3} M_{ij} C_{\text{orig},j} \quad , \tag{1a}$$

$$\tilde{C}_{\text{mix},i} = \sum_{j=1}^{3} M_{ij} C_{\text{mix},j} \quad . \tag{1b}$$

The second step accomplishes a weighted sum of these two transformed colors,

$$\tilde{C}_{\text{dst},i} = f_i(d)\tilde{C}_{\text{orig},i} + g_i(d)\tilde{C}_{\text{mix},i} \quad , \tag{2}$$

where d is some scalar distance measure depending on the distance between the camera and the object to be colored. Typical choices for this distance measure are either the z value (depth value) of the object or the Euclidean distance between camera and object (i.e., radial fog). However, even some abstract "distance" measures may be useful for specific applications, as described in Section 5. Both weight functions $f_i(d)$ and $g_i(d)$ may differ for each color component i, allowing for a distinct mixture of the two colors. This is one of the important differences compared to standard intensity depth-cueing, which has equal weights for each color component. In many applications, the weight functions are not independent of each other, but give rise to an affine combination of the two input colors, i.e.,

$$f_i(d) + g_i(d) = 1 \quad ,$$

for all valid values of d.

In the third and final step, the blended color $\mathbf{C}_{\text{dst}}$ is transformed into the original basis $\{\mathbf{P}_i\}$ by means of the inverse matrix M^{-1},

$$C_{\text{dst},i} = \sum_{j=1}^{3} M_{ij}^{-1} \tilde{C}_{\text{dst},j} \quad . \tag{3}$$

The actual color of that part of the object is then given by $(C_{\text{dst},i})$.

The mixing color is constant during the whole rendering process. Therefore, the corresponding transformation Eq. (1b) needs to be performed only once in a preprocessing step. Contrarily, Eq. (1a) has to be processed for each pixel, as its input color $\mathbf{C}_{\text{orig}}$ might change from pixel to pixel. As a consequence, Eqs. (2) and (3) have to be evaluated for each pixel as well.

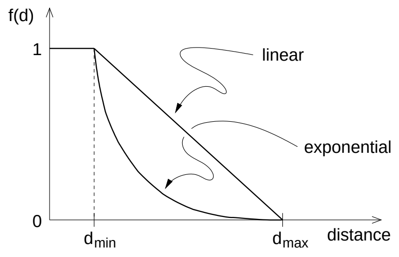

Figure 2: Linear and exponential weight functions.

4 Appropriate Parameters for the Generic Depth-Cueing Scheme

How can the above scheme facilitate color-based depth-cueing? What are appropriate choices for the intermediate basis $\{\tilde{\mathbf{P}}_i\}$, the weight functions $f_i(d)$ and $g_i(d)$, and the admixing color $\mathbf{C}_{\text{mix}}$? This section gives some useful sets of parameters for the basis vectors and weight functions and establishes the relationship between these parameters and corresponding physiologically based depth cues.

Intensity Depth-Cueing

Intensity depth-cueing can be realized by choosing appropriate parameters for our scheme. Here, colors in the foreground are unaltered; objects further away are reduced in intensity, ultimately fading away into the black background. Therefore, the admixing color has to be completely black, i.e., $(C_{\text{mix},i}) = (0,0,0)$. The same positive weight function is employed for an affine combination of each color component, i.e., $f_i(d) = f(d)$ and $g_i(d) = 1 - f(d)$. Typical choices for $f(d)$ represent a linear or exponential behavior within an interval $d \in [d_{\text{min}}, d_{\text{max}}]$, cf., for example, the OpenGL reference [34]. The bounds d_{min} and d_{max} can be specified by the user or by the 3D viewer. For ranges outside $[d_{\text{min}}, d_{\text{max}}]$, $f(d)$ is constant. Figure 2 shows a plot of linear and exponential weight functions.

For intensity depth-cueing, $f_i(d)$ is identical for all color components and therefore the weighted sum, Eq. (2), and the basis transformation, Eqs. (1) and (3), are commutative. Consequently, the basis transformations become superfluous and can be canceled. The weighted sum can be performed with respect to any basis, even in the original color system, for example RGB. This application is usually implemented by standard fogging, but can be represented in our scheme as well.

Another, almost identical application just replaces black by white as admixing color. In this way, both the intensity and saturation are changed. This approach is especially useful for illustrations on a white background. In another variation of this depth-cueing model, some additional color tone is admixed. Ebert and Rheingans [8], for example, add a bluish tone to enhance the effect of atmospheric depth-cueing. All these approaches have in common an identical weight function, independent of the color component.

Figure 4 provides a simple visualization of intensity-based depth-cueing. Image (a) represents the original yellow object. Image (b) illustrates the effect of depth-cueing by gradually changing yellow to black from bottom to top. Figure 3 shows depth-cueing for a 3D example. Image (a) depicts the original test scene, which consists of differently colored spheres. Image (b) illustrates depth-cueing by fading to black.

Saturation Depth-Cueing

The following depth-cueing approach is based on a saturation gradient and cannot be reproduced by standard fogging. Here, more

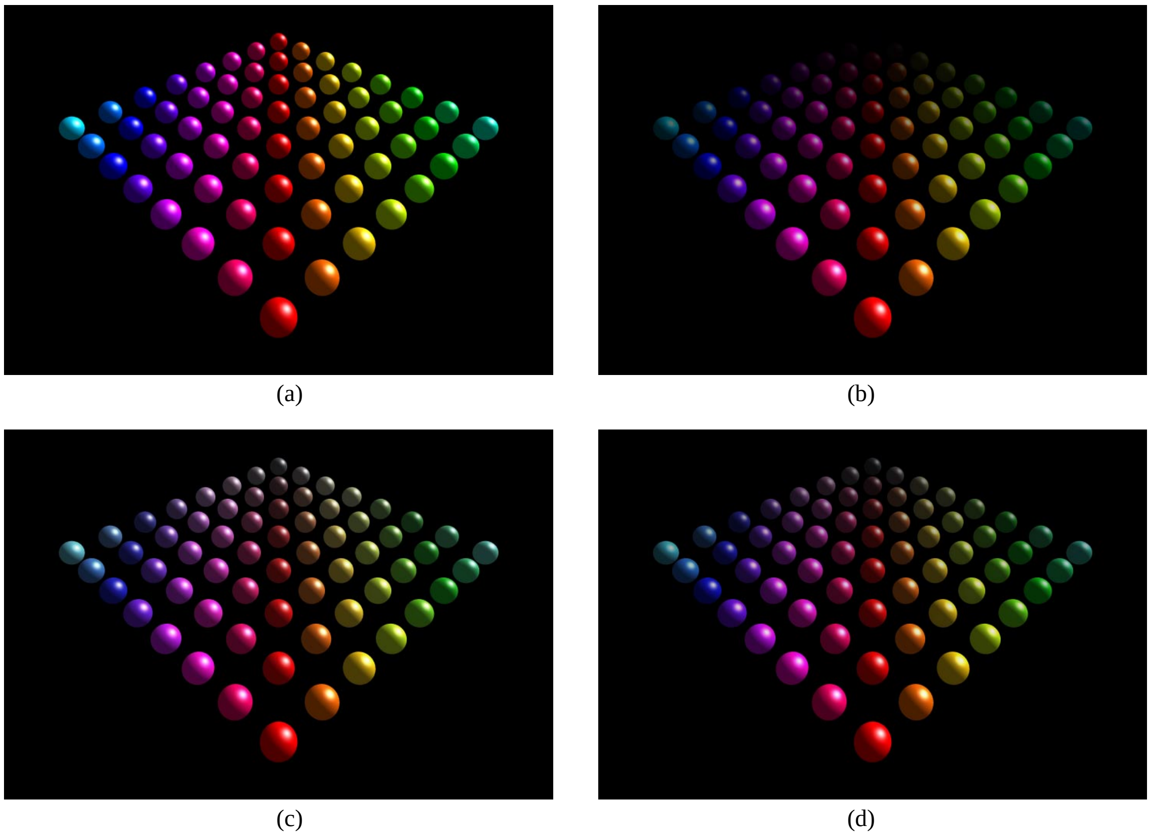

Figure 3: Comparison of different depth-cueing approaches. Image (a) shows the test scene without any depth-cueing, (b) visualizes intensity depth-cueing, (c) shows saturation depth-cueing, and (d) combines saturation and softened intensity depth-cueing.

distant objects are rendered in more desaturated colors; however, the intensities of their colors should remain constant.

Figure 5 illustrates how such a desaturation can be described geometrically in RGB tristimulus space. For clarity of the diagram, the blue axis is removed; it would be perpendicular to the red and green axes. The new color $\mathbf{C}_{\text{sat}}$ has to lie on a plane which contains the original color $\mathbf{C}_{\text{src}}$ and which is perpendicular to the gray axis. This plane is a hyperplane of isoluminant colors. In this way, the intensities of $\mathbf{C}_{\text{src}}$ and $\mathbf{C}_{\text{sat}}$ are identical. Furthermore, $\mathbf{C}_{\text{src}}$ needs to be shifted towards the gray axis, preventing any hue change. Both criteria can be met by the following construction. First a linear interpolation of the original color $\mathbf{C}_{\text{src}}$ and a color on the gray axis, $\mathbf{C}_{\text{gray}}$, is computed. The intermediate result, $\mathbf{C}_{\text{interpolation}}$, is desaturated, but shows a shift in intensity. This change in intensity is compensated by a subsequent projection of $\mathbf{C}_{\text{interpolation}}$ onto a plane perpendicular to the gray axis.

The same desaturation effect can be generated within our depth-cueing scheme. In an appropriate intermediate basis $\{\tilde{\mathbf{P}}_i\}$, one basis vector, for example $\tilde{\mathbf{P}}_1$, is collinear with the gray axis. The other two basis vectors should be chosen perpendicular to $\tilde{\mathbf{P}}_1$; $\tilde{\mathbf{P}}_2$ and $\tilde{\mathbf{P}}_3$ themselves do not need to be perpendicular. This construction ensures that intensity is only described by the coefficient of $\tilde{\mathbf{P}}_1$. Hue and saturation are given by a combination of the coefficients of $\tilde{\mathbf{P}}_2$ and $\tilde{\mathbf{P}}_3$. The admixing color $\mathbf{C}_{\text{mix}}$ is set to an arbitrary color on the gray axis, for example, to white or to black. The weight function is of the form,

$$f_1(d) = 1\,, \quad f_2(d) = f_3(d) = f(d) \quad ,$$

where $f(d)$ describes depth dependency, for example, by a linear or exponential behavior (as in the previous intensity depth-cueing approach). An affine combination of colors is employed, i.e., $g_i(d) = 1 - f_i(d)$.

By setting $f_1(d)$ identically to one, we ensure that the final color $\mathbf{C}_{\text{dst}}$ always has the same intensity as the input color $\mathbf{C}_{\text{orig}}$. By choosing gray as $\mathbf{C}_{\text{mix}}$, we make sure that the final color lies between the original color and the gray axis and we prevent any change in hue. Explicit values for the transformation matrix to and from $\{\tilde{\mathbf{P}}_i\}$ are given in Appendix A.

Figure 4 (c) provides a simple visualization of saturation-based depth-cueing. The weight function is almost identical to the weight function for intensity depth-cueing in Figure 4 (b), the only difference is that the first component is $f_1(d) = 1$. Analogously, Figure 3 (c) shows the 3D test scene with saturation-based depth-cueing. On one hand, the saturation depth cue is not as strong as the intensity depth cue, Figure 3 (b). On the other hand, a saturation gradient does not influence the intensity contrast and therefore allows to visualize even background objects. These far-away objects are not visible with intensity depth-cueing.

Hue Gradient

So far, depth cues based on brightness and saturation were presented. The third psychological term is the hue of a color. Therefore, it seems to be obvious that a hue gradient could be used as a depth cue as well. Experimental findings [28], however, indicate that a red–green hue gradient does not affect depth perception. Therefore, we do not consider a hue gradient as an application in this paper. On the other hand, we do not, by any means, want to rule out the possibility that some kind of hue gradient might support depth perception.

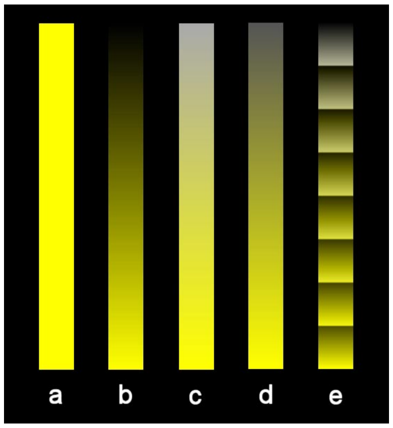

Figure 4: Comparison of color-mapping techniques. Image (a) shows the original test scene, (b) visualizes intensity depth-cueing (from bottom to top), (c) shows saturation variations, (d) combines saturation and and softened intensity changes, and (e) shows saturation changes with a superimposed brightness "sawtooth" structure.

From our point of view, hue changes make sense in applications which visualize features in a more abstract way. For example, hue changes could be determined by other means than the geometric distance, such as features or semantics of the scene, in order to visualize these features.

Combination of Depth Cues

In addition to pure intensity or saturation gradients, flexible combinations of these cues can be readily combined within our framework. Let us consider the parameters as presented above for intensity-cueing. By choosing different functions for $f_1(d)$ and for $f_2(d) = f_3(d)$, a great variety of combinations of intensity and saturation depth-cueing can now be achieved. For example, ranges and scales for the weighting functions can be different, so that saturation cues would be effective in the proximity of the camera and intensity fading would rather be employed in more remote regions.

Figure 4 (d) visualizes the combination of saturation-based and intensity-based depth-cueing. Here, the weight function is given by $f_1(d) = 1 - (1 - f(d))/2$ and $f_2(d) = f_3(d) = f(d)$, where $f(d)$ is the weight function used for intensity depth-cueing in Figure 4 (b). Figure 3 (d) shows the same depth cue for the 3D test scene. The weight function has parameters analogous to those in Figure 3 (b). In Figure 3 (d), the dimming effect is only half of the desaturation effect, combining the advantages of both approaches: The depth cue is stronger than in mere saturation-cueing and, in contrast to intensity-cueing, background objects are still visible.

Figure 4 (e) shows an example of a quite complex combination of intensity and saturation cues. A saturation gradient as of Figure 4 (c) is superimposed by a "sawtooth" function for intensity depth-cueing, i.e., the weight function $f_1(d)$ has recurrent maximum and minimum values that are linearly interpolated. The resulting image allows to identify areas of equal distance from the camera. In this way, depth-cueing can even facilitate quantitative distance measures. In the following section, we illustrate how this choice of parameters supports depth perception in terrain visualization.

5 Applications

In this section, we demonstrate the aptitude of the color-based depth-cueing scheme by means of various applications in non-

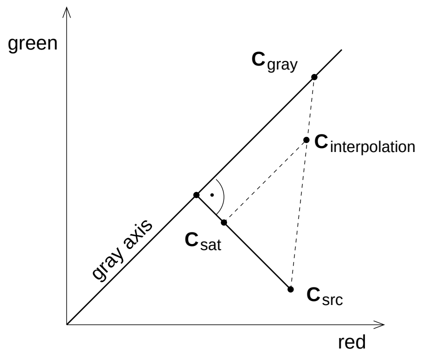
green
red
gray axis
$\mathbf{C}_{\text{gray}}$
$\mathbf{C}_{\text{interpolation}}$
$\mathbf{C}_{\text{sat}}$
$\mathbf{C}_{\text{src}}$

Figure 5: Change of saturation. The new color $\mathbf{C}_{\text{sat}}$ can be produced by a weighted sum of the original color, $\mathbf{C}_{\text{src}}$, and a color on the gray axis, $\mathbf{C}_{\text{gray}}$, followed by a subsequent projection onto a plane perpendicular to the gray axis.

photorealistic rendering and visualization. These examples are, of course, not exhaustive, but show what type of applications can benefit from color-based depth cues.

Figure 6 demonstrates depth-cueing for the non-photorealistic rendering of technical illustrations. Figure 6 (a) shows the original engine block. In Figure 6 (b), cool-to-warm tone shading [10] is applied to enhance the recognition of surface orientation. A cool (blue) to warm (tan) transition of color tones indicates a change of the surface normal from left to right. Figure 6 (c) illustrates cool/warm shading and saturation depth-cueing and Figure 6 (d) illustrates cool/warm shading and intensity depth-cueing. The main differences between Figures 6 (b)–(d) appear in furthermost parts of the engine, especially in the upper portion of the images. Saturation depth-cueing causes rather subtle color changes, whereas intensity depth-cueing harshly affects the image by completely fading away parts of the engine. This example demonstrates that saturation depth-cueing is close to Tufte's strategy of the smallest effective difference, making visual distinctions as subtle as possible, but still clear and effective. In particular, technical illustrations benefit from saturation depth-cueing because all structures—even in the background—are retained.

Figure 7 shows a typical application in information visualization. Here, a large amount of hierarchically structured data is given in the form of a mathematical tree and visualized by a so-called cone tree [24]. The elements of the tree are represented by boxes and cones and connected by thin lines. These graphical objects are located and viewed in 3D space. Therefore, depth perception is quite important for the understanding of the tree structure. The original rendering in Figure 7 (a) includes only the standard depth cues in the form of perspective and occlusion. With the saturation depth cue being enabled in Figure 7 (b), the depth structure becomes much clearer. The top and bottom images in the center of Figure 7 show magnified details of the respective renderings. The in-between image provides a comparison between these two detail images, showing the differences of saturation levels of the two original images as gray-scale values.

Figure 8 demonstrates depth-cueing for terrain rendering. Figure 8 (a) shows the original scene, Figure 8 (b) shows the scene with saturation depth-cueing, and Figure 8 (c) shows the scene with intensity depth-cueing. In all the images of Figure 8, depth-cueing is only applied to the terrain itself, the background sky and ocean keep their original colors. In Figure 8 (d), saturation depth-cueing is combined with a "sawtooth" function for intensity depth-cueing, analogously to the parameters for Figure 4 (e). The resulting image allows to identify lines of equal distance from the camera. In this way, depth-cueing even facilitates quantitative distance measures.

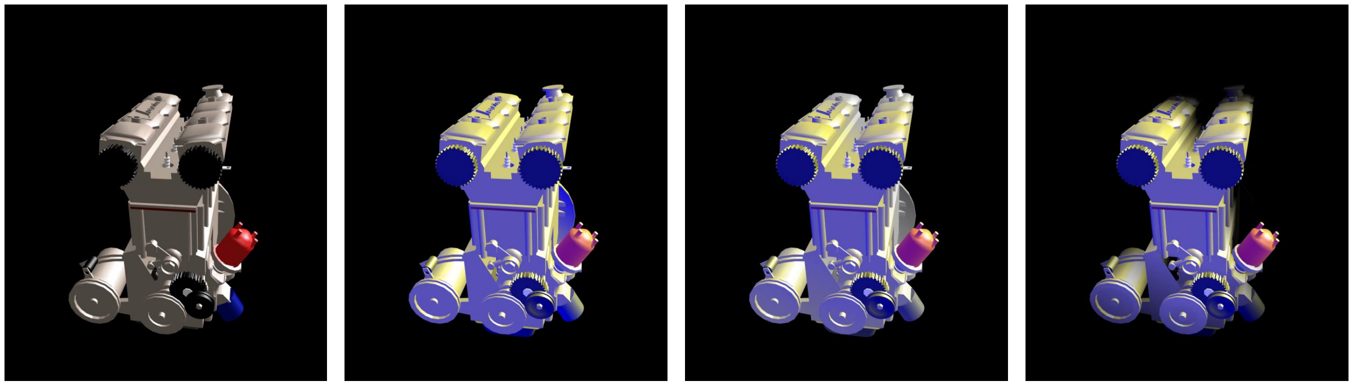
(a) (b) (c) (d)

Figure 6: Non-photorealistic rendering of an engine block. Image (a) shows the original scene, (b) cool/warm shading without depth-cueing, (c) cool/warm shading with saturation depth-cueing, and (d) cool/warm shading with intensity depth-cueing.

This technique is related to classical contour lines (isolines) in topographical relief maps, but is more subtle. It is similar to enridged contour maps [32] used for height field visualization. It is also influenced by works by the Flemish painter Jan Brueghel the Elder, who enhances depth perception by alternate darker and lighter areas, as in [2]. The "sawtooth" approach is especially useful for extended objects like terrains. However, it is less suited for scenes consisting of many small and separate objects because the context of lines of equal distance is destroyed by the blank regions between these objects.

In Figure 9, saturation depth-cueing is applied to an abstract "distance" measure. Figure 9 (a) shows a standard 3D view on a chess board. In Figures 9 (b) and (c), however, some of the chessmen are accentuated by using highly saturated colors, whereas the less important objects take a back seat caused by desaturated colors. Figure 9 (b) shows an extreme case: The highlighted objects are fully saturated and the "background" objects are only in gray-scale. Here, the weight functions $f_{2,3}(d)$ take only the two extreme values 1 and 0 for the "foreground" and the "background" objects, respectively. In the framework of depth-cueing, two different depth values are assigned to the two groups of objects, depending on their semantic value. In this approach, the "depth" value is determined by the user or some other outside source, but not by purely geometric quantities. We call this type of visualization *semantic depth-cueing*, similarly to the techniques by Kosara et al. [18] for semantic depth-of-field.

Figure 9 (c) illustrates that semantic depth-cueing can be combined with geometric depth-cueing. The "foreground" objects are still completely saturated. The saturation of the "background" objects, however, is controlled by their Euclidean distance to the camera. On one hand, this combined approach conveys more information than mere semantic depth-cueing. On the other hand, the mapping from saturation value to semantic value is no longer unique, so that understanding might be complicated. Therefore, the benefit of the variations of semantic depth-cueing largely depends on the available colors, semantics, and depth structure of the scene.

As another application, depth-cueing can be applied to non-photorealistic volume rendering. Ebert and Rheingans [8] use a standard fogging approach for depth-cueing in volume illustrations, which could be replaced by our flexible depth-cueing scheme.

A further field of application is in image-based rendering (IBR). Recent work [5, 3, 26] deals with real-time processing and rendering of image-based representations by exploiting graphics hardware. Sophisticated depth cues can improve the perception of spatial structures. As a closely related application, range images could also be visualized.

In a completely different setting, saturation control is required in virtual reality facilities. In an immersive virtual environment like a CAVE, luminance values are very low because projection devices are not able to produce very bright images and stereo glasses further reduce intensity. Very often the user is limited to viewing conditions in the mesoscopic range, where color perception by the human visual system is significantly reduced [27]. Therefore, colors are perceived in low saturation in such VR environments. Our depth-cueing scheme allows to increase the saturation of colors by supplying negative values and values larger than 1 in the weight functions, i.e., by shifting colors further away from the gray axis. In this way, the saturation of the displayed colors can be adjusted to the viewing conditions in real time.

6 Realization on Graphics Hardware

The presented depth-cueing scheme is quite flexible and allows many different types and variations of color depth cues. Nevertheless, the required arithmetic operations are—computationally—not very demanding and low in number. Only a few multiplications and summations are necessary and, for example, no division is needed. Therefore, the depth-cueing scheme can be implemented completely on contemporary consumer graphics hardware. In this context, we exploit programmable and configurable parts of the graphics pipeline, in particular, the transform and lighting portion and flexible fragment blending operations.

The depth-cueing scheme can be implemented transparently to the user or programmer of a graphics viewer. Our example implementation runs on nVidia's GeForce 3 and, based on OpenGL 1.2, uses the `register_combiners`, `vertex_program`, and `multitexture` extensions. All required operations are also available in the form of similar extensions on ATI's Radeon 8500, providing hardware support for sophisticated depth-cueing on this card as well.

A standard renderer just needs to be extended by an additional initialization step which loads a vertex and a register combiner program described below. Our implementation is based on the *OpenSceneGraph* API [21] and requires only minimal extensions to the standard renderer. Moreover, the implementation comes with (almost) no additional rendering costs and therefore allows interactive applications.

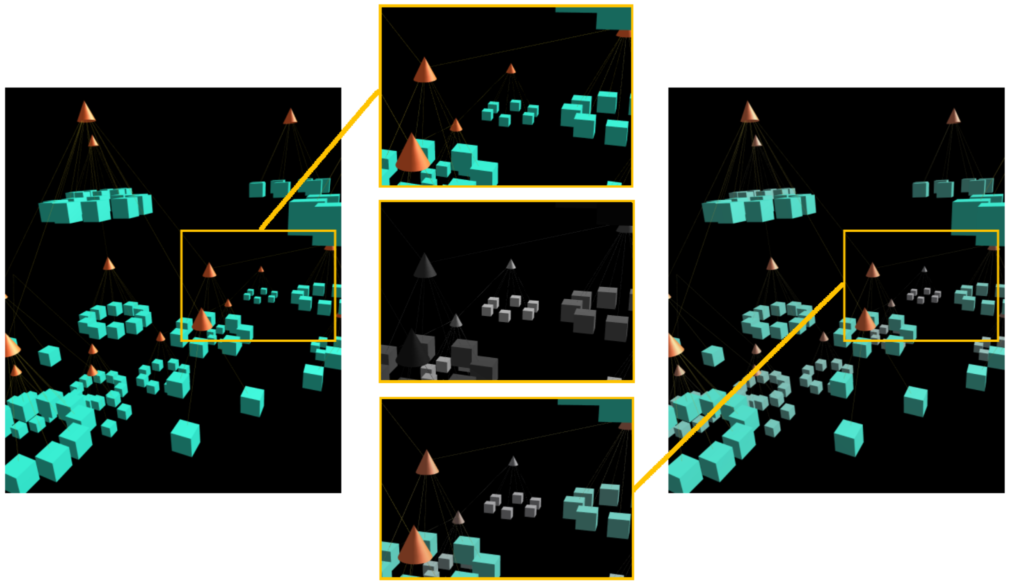
(a) (b)

Figure 7: Cone tree representation in information visualization. The left image shows the original scene; in the right image, saturation depth-cueing is applied to the same scene. The top and bottom images in the middle show magnified details; the central image visualizes the difference between the two detail images.

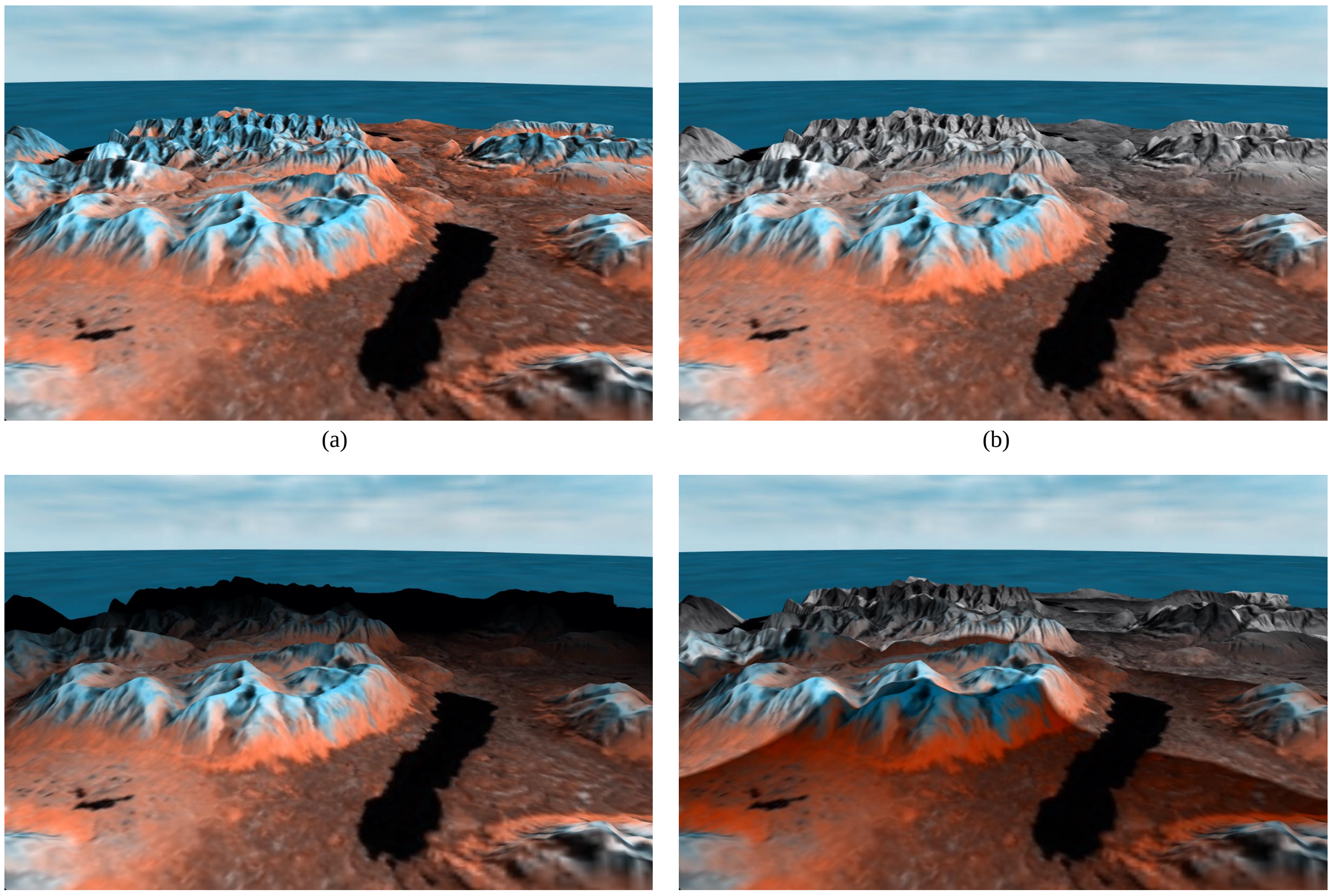
(a) (b) (c) (d)

Figure 8: Terrain rendering. Image (a) shows the original scene, (b) uses saturation depth-cueing, (c) uses intensity depth-cueing, and (d) combines saturation depth-cueing with "sawtooth" intensity depth-cueing.

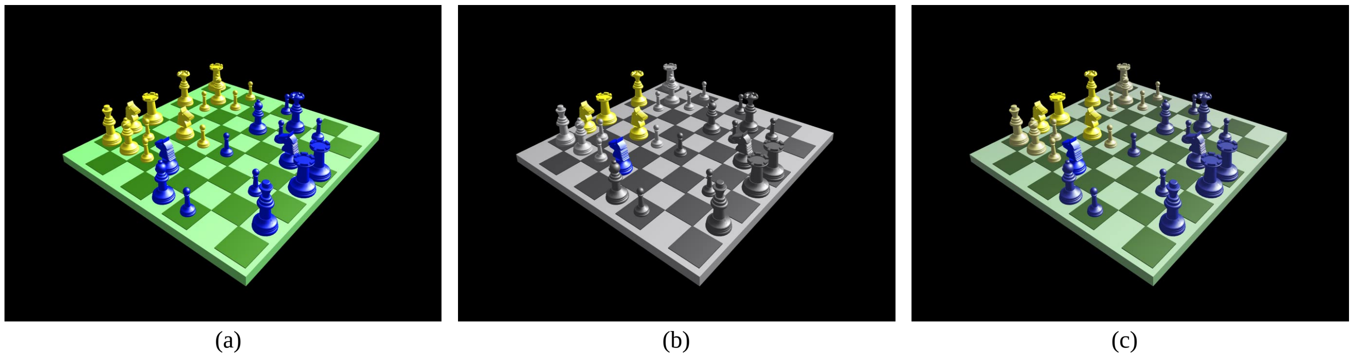

Figure 9: Semantic depth-cueing. Image (a) shows a standard 3D view on a chess board. In image (b), some chessmen are emphasized by highly saturated colors. Image (c) illustrates a combination of semantic and geometric depth-cueing.

The only assigned task for the vertex program—besides standard transform and lighting calculations—is to determine the depth value d of a vertex and set the texture coordinate of a one-dimensional texture to this depth value. For the calculation of the Euclidean (radial) distance between camera and vertex, for example, only four additional vertex program commands are necessary. Even simpler is the computation of the standard depth value, i.e., the distance perpendicular to the image plane. The cool-to-warm tone shading model for Figure 6 is also implemented in the vertex program, providing a single pass rendering for both the diffuse cool/warm terms and the white Phong highlights.

The next change affects the texture fetching unit. Here, one additional texture lookup for the three components, $f_i(d)$, of the weight function has to be included if an affine combination of colors is implemented. If $f_i(d)$ and $g_i(d)$ are independent of each other, one further lookup for $g_i(d)$ is required. Multiple texture lookups are supported by modern graphics hardware; four or even more texture units are provided without need for multi-pass rendering. Note that, for the GeForce 3, texture fetches are the only possible way to attach some kind of depth information to fragments that are processed in the subsequent pixel blending stage. Unfortunately, the GeForce 3 does not provide access to the depth value or fogging factor in this part of the rendering pipeline. On the other hand, our texture approach gives much more flexibility in the choice of blending functions than a simple access to the depth value could offer, since any kind of weighting functions can be represented as sampled texture. In particular, non-linear depth functions, such as exponential functions, are correctly reproduced on a per-pixel basis.

Finally, the pixel blending unit (register combiner stage) is responsible for the main task, i.e., the transformation and blending of a fragment color according to Eqs. (1a), (2), and (3). Figure 10 illustrates the structure of the corresponding configuration of register combiners for the implementation on the GeForce 3. All combiner stages merely process the RGB portions of a fragment and do not change the alpha portion.

The first four general combiners are used to transform the input color $(C_{\mathrm{orig},i})$ from RGB color space to the intermediate basis $\tilde{\mathbf{P}}_i$. In the diagram, the input color is represented by `primary color`. Combiners 0 and 1 compute dot products between the input color and the three rows of the transformation matrix M, i.e., each of these dot products contains one of the three components of the transformed color. Unfortunately, the result of a dot product is stored in all components of the output color. Therefore, combiners 2 and 3 compute component-wise multiplication of the results of the previous dot products with weights $(1,0,0)$, $(0,1,0)$, and $(0,0,1)$, respectively. In this way, wrong entries are masked out and the final transformed color is obtained by a sum of the results of these three multiplications. This summation is also performed in combiners 2 and 3.

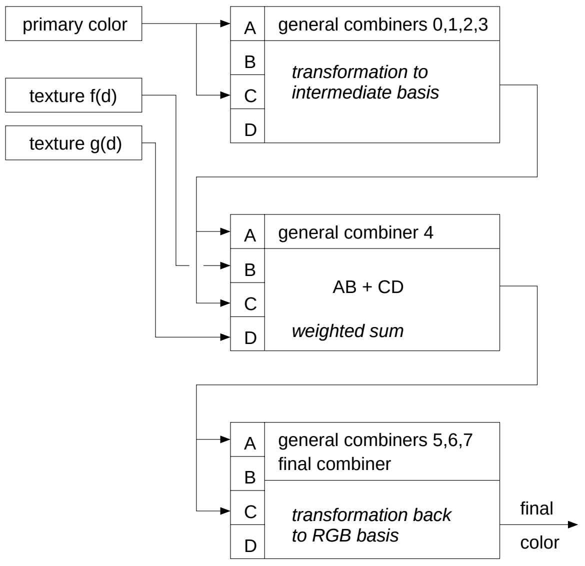

Figure 10: Configuration of register combiners for the transformation and blending of a fragment color.

Combiner 4 implements the weighted sum according to Eq. (2). It takes one or two texture fragments as input, containing values for $f_i(d)$ or $g_i(d)$, respectively. The transformed color from combiner 3 is another input parameter and undergoes a component-wise multiplication with $f_i(d)$, yielding the first term of the weighted sum in Eq. (2). The last input port takes the transformed admixing color $\tilde{C}_{\mathrm{mix},i}$, which is a constant color specified by the user. In a second component-wise multiplication, the second part of the weighted sum is computed. Finally, the output of combiner 4 contains the sum of both multiplication, i.e., the complete weighted sum from Eq. (2). If an affine combination of colors is intended, the second texture $g_i(d)$ is redundant and the corresponding input port takes $1 - f_i(d)$, which can be realized by an input mapping `unsigned_invert`.

General combiners 5,6,7, and the final combiner implement the transformation of the blended color back to RGB color space according to Eq. (3). The configuration is similar to combiners 0–3; essentially, the transformation matrix M is replaced by the inverse matrix M^{-1}. Ultimately, the final combiner provides the intended color $\mathbf{C}_{\mathrm{dst}}$ in RGB color space.

Note that the matrices M and M^{-1} may have negative elements.

As rows of these matrices have to be transferred to register combiners, an input mapping of the form `expand_normal` may be necessary. In this way, parameters are mapped from the interval $[0,1]$ to $[-1,1]$, allowing negative entries. Another caveat is the limited resolution and possible range of values in the register combiner pipeline of the GeForce 3. Therefore, the matrix M should be chosen in a way that allows both its matrix elements and the components of the transformed color to lie within [-2,2].

All the example images of this paper and for the accompanying video were generated by our hardware-based implementation. The frame rates for the depth-cueing application are identical to those of the original *OpenSceneGraph* viewer for all example scenes. This shows that the additional operations in the transform and lighting part and the register combiner unit do not impair rendering performance in typical applications. Only for scenes with extreme rasterization demands—such as texture-based volume rendering—the additional operations in the register combiner unit could decrease the overall frame rate, depending on the actual properties of the graphics chip.

In the remaining part of the section, we present another approach to hardware-based color depth-cueing. The basic idea is to detach the first color transformation step from the register combiners. In fact, this transformation to the intermediate color basis, according to Eq. (1a), could also be implemented in a vertex program. As vertex programming allows even more complex and a much larger number of operations than register combiners, original colors $\mathbf{C}_{\mathrm{orig}}$ can easily be transformed on a per-vertex basis. Rasterization is always based on a triangulated surface; and colors are interpolated linearly within a triangle during scanline conversion. As linear interpolation and a linear transformation via a matrix are commutative, colors may very well be transformed before scanline conversion. In this way, the rasterization stage could be unburdened and many unnecessary calculations could be avoided—at the cost of only three transformations per triangle within the transform and lighting part of the rendering pipeline.

On current consumer graphics hardware, however, this approach is not yet feasible. Major problems are caused by a limited range and resolution of color values on the data path from the transform and lighting stage to the rasterization unit. In particular, colors are clamped to values from $[0,1]$. However, negative values and values larger than one may occur. As a solution to this problem, colors could be mapped to $[0,1]$ at the end of the transform and lighting stage and remapped to the original interval at the beginning of the rasterization stage. With a limited color depth of usually eight bits, this causes unacceptable color artifacts. As this is not a fundamental problem, this kind of implementation might become feasible in the near future on improved chips with color channels of higher resolution or wider value range.

7 Conclusion

In this paper, we have proposed a generic and widely applicable color-based depth-cueing scheme. It rests upon the fundamental principle of trichromatic generalization of human color perception. Arbitrary hyperplanes in the 3D tristimulus space of colors can be specified, and different ways of combining colors can be applied to color differences in these planes and perpendicular to them.

We have derived parameters for this generic scheme to allow for well-known intensity depth-cueing and rather neglected saturation depth-cueing. In addition, we have presented variations and combinations of these core approaches to further improve depth perception for different fields of application. Semantic depth-cueing has been introduced as an abstract depth-cueing approach. Typical fields of application comprise technical illustrations, artistic rendering, information visualization, terrain visualization, volume illustrations, or image-based rendering.

Although our depth-cueing scheme is very flexible and powerful, it requires only a few and simple computational operations—just two linear transformations of 3D vectors and a weighted sum of two vectors. All of the operations are supported by contemporary consumer graphics hardware in single-pass rendering. The implementation of our scheme needs only a few configurations of the rasterization as well as transform and lighting units and it is transparent to standard rendering. Existing applications can be easily enriched by sophisticated depth-cueing without performance cutbacks. Real-time capability is extremely useful, since it allows color-based depth cues to be combined with other depth cues—such as motion parallax, active movement, perspective, and occlusion—in order to improve depth perception significantly.

A Transformation to Gray Axis

For saturation depth-cueing as described in Section 4, we have to find an appropriate intermediate basis $\{\tilde{\mathbf{P}}_i\}$ in which one basis vector, for example $\tilde{\mathbf{P}}_1$, is collinear with the gray axis. The other two basis vectors, $\tilde{\mathbf{P}}_{2,3}$, have to be perpendicular to $\tilde{\mathbf{P}}_1$.

For our implementation, we assume that the gray axis stretches from black to white in RGB space, i.e., from $(0,0,0)$ to $(1,1,1)$. Therefore, the gray vector $\tilde{\mathbf{P}}_1$ is the sum of the RGB basis vectors, up to an arbitrary scaling factor α, i.e., $\tilde{\mathbf{P}}_1 = \alpha\sum_{i=1}^{3}\mathbf{P}_i$. The other two basis vectors are chosen perpendicular to $\tilde{\mathbf{P}}_1$ and to each other. One possible choice for the transformation matrix is

$$M_{ij} = \frac{1}{3}\begin{pmatrix} 1 & -1+\sqrt{3} & -1-\sqrt{3} \\ 1 & -1-\sqrt{3} & -1+\sqrt{3} \\ 1 & 2 & 2 \end{pmatrix} \quad .$$

The transformation back to RGB color space is based on the inverse matrix M^{-1}.

The maximum and minimum values of M_{ij} are $1/3$ and $(-1-\sqrt{3})/3$, respectively. For the implementation on the GeForce 3, the matrix elements have to be mapped from $[-1,1]$ to $[0,1]$ before they can be transferred as parameters to the register combiner stage. The respective input mapping for the register combiners has to be `GL_EXPAND_NORMAL_NV` in order to remap the matrix elements back to the true values.

Other choices of matrices are also possible. In particular, the definition of the luminous vector $\mathbf{Y}$ from the CIE XYZ color space could serve as a basis for the transformation matrix.

References

[1] M. J. Bailey and D. Clark. Using ChromaDepth to obtain inexpensive single-image stereovision for scientific visualization. *Journal of Graphics Tools*, 3(3):1–9, 1998.

[2] J. Brueghel the Elder. Christ preaching at the seaport. Oil on wood, Alte Pinakothek, Munich, 1598.

[3] C. Buehler, M. Bosse, L. McMillan, S. J. Gortler, and M. F. Cohen. Unstructured lumigraph rendering. In *SIGGRAPH 2001 Conference Proceedings*, pages 425–432, 2001.

[4] C. J. Curtis, S. E. Anderson, J. E. Seims, K. W. Fleischer, and D. H. Salesin. Computer-generated watercolor. In *SIGGRAPH 1997 Conference Proceedings*, pages 421–430, 1997.

[5] P. E. Debevec, Y. Yu, and G. D. Borshukov. Efficient view-dependent image-based rendering with projective texture-mapping. In *Eurographics Rendering Workshop '98*, pages 105–116, 1998.

[6] O. Deussen, S. Hiller, C. van Overveld, and T. Strothotte. Floating points: A method for computing stipple drawings. In *Computer Graphics Forum (Eurographics 2000)*, volume 19(3), 2000.

[7] D. Ebert, K. Musgrave, D. Peachey, K. Perlin, and S. Worley. *Texturing and Modeling*. Academic Press, San Diego, 1998.

[8] D. Ebert and P. Rheingans. Volume illustration: Non-photorealistic rendering of volume models. In *IEEE Visualization 2000 Proceedings*, pages 195–202, 2000.

[9] J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes. *Computer Graphics — Principles and Practice*. Addison-Wesley, 2nd edition, 1992.

[10] A. Gooch, B. Gooch, P. Shirley, and E. Cohen. A non-photorealistic lighting model for automatic technical illustration. In *SIGGRAPH 1998 Conference Proceedings*, pages 101–108, 1998.

[11] B. Gooch and A. A. Gooch. *Non-Photorealistic Rendering*. A. K. Peters, Natick, Mass., 2001.

[12] B. Gooch, P.-P. Sloan, A. Gooch, P. Shirley, and R. Riesenfeld. Interactive technical illustration. In *1999 ACM Symposium on Interactive 3D Graphics*, pages 31–38, 1999.

[13] S. Green, D. Salesin, S. Schofield, A. Hertzmann, P. Litwinowicz, A. A. Gooch, C. Curtis, and B. Gooch. *SIGGRAPH 1999 Course 17:* Non-photorealistic rendering, 1999.

[14] C. G. Healey, V. Interrante, and P. Rheingans. *SIGGRAPH 1999 Course 6:* Fundamental issues of visual perception for effective image generation, 1999.

[15] G. S. Hubona, P. N. Wheeler, G. W. Shirah, and M. Brandt. The relative contributions of stereo, lighting, and background scenes in promoting 3D depth visualization. *ACM Transactions on Computer-Human Interaction*, 6(3):214–242, Sept. 1999.

[16] V. Interrante, P. Rheingans, J. Ferwerda, R. Gossweiler, and C. G. Healey. *SIGGRAPH 1998 Course 32:* Applications of visual perception in computer graphics, 1998.

[17] C. A. Kelsey. Detection of visual information. In W. R. Hendee and P. Wells, editors, *Perception of Visual Information*, pages 30–51, Wien, 1993. Springer.

[18] R. Kosara, S. Miksch, and H. Hauser. Semantic depth of field. In *IEEE Symposium on Information Visualization 2001 (InfoVis 2001)*, 2001.

[19] L. Markosian, M. A. Kowalski, S. J. Trychin, L. D. Bourdev, D. Goldstein, and J. F. Hughes. Real-time nonphotorealistic rendering. In *SIGGRAPH 1997 Conference Proceedings*, pages 415–420, 1997.

[20] S. W. Mereu and R. Kazman. Audio enhanced 3D interfaces for visually impaired users. In *Proceedings of ACM CHI 96 Conference*, volume 1, pages 72–78, 1996.

[21] Open Scene Graph. Web page. http://www.openscenegraph.org, 2001.

[22] E. Praun, H. Hoppe, M. Webb, and A. Finkelstein. Real-time hatching. In *SIGGRAPH 2001 Conference Proceedings*, pages 579–584, 2001.

[23] R. Raskar. Hardware support for non-photorealistic rendering. In *ACM SIGGRAPH/Eurographics Workshop on Graphics Hardware*, pages 41–46, 2001.

[24] G. G. Robertson, J. D. Mackinlay, and S. K. Card. Cone trees: Animated 3D visualizations of hierarchical information. In *Proceedings of ACM CHI'91 Conference on Human Factors in Computing Systems*, pages 189–194, 1991.

[25] M. P. Salisbury, S. E. Anderson, R. Barzel, and D. H. Salesin. Interactive pen-and-ink illustration. In *SIGGRAPH 1994 Conference Proccedings*, pages 101–108, 1994.

[26] H. Schirmacher, L. Ming, and H.-P. Seidel. On-the-fly processing of generalized lumigraphs. *Computer Graphics Forum*, 20(3), 2001.

[27] F. Schöffel, W. Kresse, S. Müller, and M. Unbescheiden. Do IPT systems fulfill application requirements? – A study on luminance on large-scale immersive projection devices. In *Proceedings of the 3rd International Immersive Projection Technology Workshop (IPT '99)*, pages 281–292. Springer, 1999.

[28] T. Troscianko, R. Montagnon, J. L. Clerc, E. Malbert, and P.-L. Chanteau. The role of colour as a monocular depth cue. *Vision Research*, 31(11):1923–1930, 1991.

[29] E. R. Tufte. *Visual Explanations: Images and Quantities, Evidence and Narrative*. Graphics Press, Cheshire, Conn., 1997.

[30] J. Tumblin and J. A. F. (eds.). Issue on *applied perception*. *IEEE Computer Graphics and Applications*, 21(5):20–85, 2001.

[31] J. van Eyck. The virgin of chancellor Rolin. Oil on wood, Musée du Louvre, Paris, 1435.

[32] J. J. van Wijk and A. Telea. Enridged contour maps. In *IEEE Visualization 2001 Proceedings*, pages 69–74, 2001.

[33] G. Winkenbach and D. H. Salesin. Computer-generated pen-and-ink illustration. In *SIGGRAPH 1994 Conference Proceedings*, pages 91–100, 1994.

[34] M. Woo, J. Neider, T. Davis, and OpenGL Architecture Review Board. *OpenGL Programming Guide*. Addison-Wesley, 1997.

[35] G. Wyszecki and W. S. Stiles. *Color Science*. John Wiley & Sons, New York, second edition, 1982.

[36] S. Zhai, W. Buxton, and P. Milgram. The partial-occlusion effect: Utilizing semitransparency in 3D human-computer interaction. *ACM Transactions on Computer-Human Interaction*, 3(3):254–284, Sept. 1996.