Luminance calibration of virtual reality displays in Unity

Richard F. Murray Department of Psychology and Centre for Vision Research, York University, Toronto, Ontario, Canada

Khushbu Y. Patel Department of Psychology and Centre for Vision Research, York University, Toronto, Ontario, Canada

Emma S. Wiedenmann Department of Psychology, Carl von Ossietzky University of Oldenburg, Oldenburg, Germany

Virtual reality (VR) displays are an increasingly popular medium for experiments on visual perception. This presents the challenge of showing precisely controlled stimuli on devices that were not primarily designed for research. Here we describe methods for controlling stimulus luminance in VR experiments created in Unity using the Built-in Render Pipeline. We discuss the Gamma/Linear setting, measuring luminance in a VR headset, and using color grading in Unity's Post-Processing Stack to make stimulus luminance proportional to achromatic RGB value. We provide MATLAB code that uses luminance measurements from a VR headset to generate the lookup table that Unity requires for linearizing luminance. We emphasize that when creating experiments in this complex environment, it is important to experiment with the rendering process to confirm that stimuli are displayed as expected. We show results of several such tests and provide code as a starting point for readers who wish to run further tests related to their own research.

Introduction

Virtual reality (VR) devices have existed in various forms for many years, but a recent wave of consumer models has made this medium newly accessible and easy to use (Greengard, 2019). For vision researchers, VR provides the flexibility of computer-generated displays while also simulating to some extent the immersive, multimodal, and interactive nature of physical reality. This makes new kinds of experiments possible but also brings new challenges. Some challenges are broad, such as understanding the extent to which visual perception in VR is similar to perception in real environments. Others are more practical, such as developing methods of controlling key stimulus properties.

Here we address a problem that is limited in scope but crucial for many experiments: how to luminance-calibrate a VR headset in the Unity environment (version 2020.1.5f1; Unity Technologies, 2020). Luminance is a fundamental stimulus property, and so is contrast, which is determined by the luminance profile of a stimulus. Luminance and contrast have dramatic effects on human performance in a wide range of tasks, and it is important to be able to control them in perceptual experiments. However, recent papers using VR methods in vision research have typically paid little attention to the luminance or colorimetric properties of stimuli. Two exceptions are Toscani et al. (2019) and Rodriguez et al. (2022), who reported color calibration methods for an HTC Vive display in the Unreal Engine environment. They found that with a suitable configuration of rendering software, the apparatus met several assumptions behind standard color calibration methods, such as color channel additivity (Brainard et al., 2002). Furthermore, they found that stimulus luminance was proportional to rendered RGB values. Another example of attention to calibration issues is Diaz-Barrancas et al. (2021), who showed that color calibration can be carried out in Unity's High-Definition Render Pipeline by integrating a standard color calibration model (Brainard et al., 2002), written in custom software, into the render pipeline.

We describe methods for controlling luminance in Unity's Built-in Render Pipeline, which is a widely used environment for developing VR experiments. Considering a specific software environment allows us to make concrete suggestions, and we also provide software tools in the Supplementary Material (https://doi.org/10.17605/OSF.IO/RYTE5). Furthermore, several of the issues we address also have wider relevance (e.g., we expect that the methods we describe can be extended to control additional stimulus properties such as color). The focus of this article is on software, and we do not address hardware-specific issues, such as differences between models of VR headsets.

Citation: Murray, R. F., Patel, K. Y., & Wiedenmann, E. S. (2022). Luminance calibration of virtual reality displays in Unity. *Journal of Vision*, *22*(13):1, 1–9, https://doi.org/10.1167/jov.22.13.1.

https://doi.org/10.1167/jov.22.13.1 Received February 18, 2022; published December 1, 2022 ISSN 1534-7362

Unity is a powerful environment for creating interactive experiments, but a recurring theme of this article is that it was primarily developed for creating games, not for research. As a result, it is important to experiment with the rendering process, to test whether rendered images are what we expect them to be. We provide several examples of such experiments and include supporting code as a starting point for further tests. Most of the information about Unity that we report here is the result of experimentation, so we believe it will be novel and useful to other researchers. We frame this article as addressing calibration issues in VR, but the methods we describe can also be used to control luminance in Unity experiments on traditional flat-panel displays.

Calibration steps

We start with a point-by-point summary of the procedure we describe below for calibration, in order to give an overview of the calibration strategy.

1. Confirm that the Unity project is in Gamma rendering mode, which is the default mode in recent versions of Unity. If it is not, then put it into Gamma mode.
2. Measure the luminance generated in the uncalibrated VR headset by a series of achromatic RGB values. These data should follow an expansive nonlinearity, like the red data points in Figure 3a. (Throughout this article, we use the term "RGB value" to denote the three integers in the range 0 to 255 that represent a pixel in video memory (as in the present case), or that specify the color of a Unity object; these are distinct from the luminance or colorimetric coordinates of the light generated by the corresponding pixels on the physical display.)
3. Use these measurements to generate a LUT texture image (a PNG file), using the MATLAB code provided in the Supplementary Material.
4. Configure Unity's Post-Processing Stack (see Supplementary Material, Section 1(f)), and load the LUT texture created in the previous step into the Color Grading component.
5. Stimulus luminance on the headset should now be proportional to achromatic RGB value. Confirm this by again measuring the luminance generated in the headset by a series of achromatic RGB values. These data should be linear, like the green data points in Figure 3a.

Unity settings

The Gamma/Linear setting

Unity has hundreds of settings, and one challenge is finding a good configuration for experiments.

Figure 1. The scene used to examine rendering in Gamma and Linear modes. Here, the white Lambertian test square is shown at a slant of 30° relative to the camera. The illumination is an achromatic, directional light source directly behind the camera. The surrounding blue background does not contribute to the lighting.

The Gamma/Linear mode setting is one important parameter. The Unity documentation provides some information about this setting (Unity User Manual, 2020, https://docs.unity3d.com/2020.1/Documentation/Manual/LinearLighting.html), but to investigate it more thoroughly, we ran some tests of the rendering process.

We created a scene where a Lambertian[1] test patch was rotated through a range of angles relative to a distant, directional light source, and at each angle, we recorded the RGB value of the rendered test patch (Figure 1). We provide the code as Supplementary Material. For a Lambertian surface, the rendered RGB value should ideally be proportional to the cosine of the angle between the surface normal and the lighting direction, for angles in the range $[-90°, 90°]$ (Pharr et al., 2017). Figure 2 shows that this is true in Gamma mode (panel a) but not in Linear mode (panel b), where the RGB values appear to have been transformed by a compressive nonlinearity. We reasoned that this nonlinearity might be the inverse of the sRGB transfer function that approximately characterizes many monitors, and Figure 2b confirms that if we transform the Linear mode RGB values by the sRGB transfer function (approximately[2] $f(x) \sim x^{2.2}$), they fall into the expected cosine pattern.

Thus, in Gamma mode, the displayed RGB value is proportional to the result of a Lambertian shading calculation, and the displayed luminance is subject to

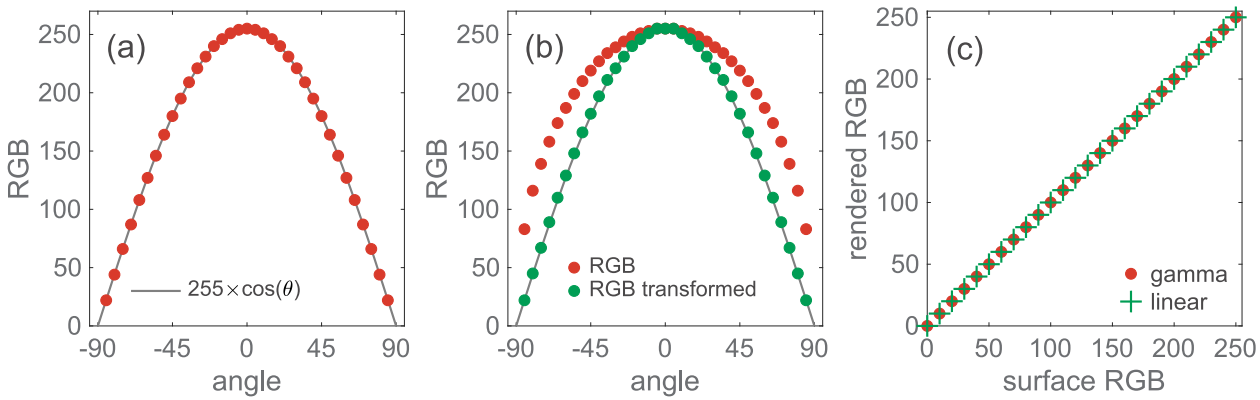

Figure 2. (a) Rendered RGB value of an achromatic Lambertian surface as a function of angle relative to light source, in Gamma mode. (b) Corresponding results in Linear mode and results transformed by the sRGB gamma function. (c) Rendered RGB value of an achromatic Lambertian surface as a function of its assigned RGB color, in Gamma mode and Linear mode. The surface was rendered under directional light with unit intensity, in the direction of the surface normal.

the gamma function of the VR headset. In Linear mode, the result of the shading calculation is transformed by an inverse gamma function $f^{-1}(x) \sim x^{-2.2}$, so that if the display has the anticipated gamma function $f(x) \sim x^{2.2}$, then the two transformations cancel and the luminance displayed is proportional to the physically correct values for the rendered scene.

We recommend using the Gamma setting (which is the default in recent versions of Unity) and correcting for the VR headset's luminance nonlinearity using the color grading method described below. One reason for taking this approach is that, if stimulus calibration is important, then it should be based on measurements from individual displays instead of industry standards such as sRGB. Another reason is that in Linear mode, Unity does not consistently apply the inverse gamma transformation to all stimuli. Figure 2c shows the rendered RGB value of an achromatic Lambertian surface whose RGB color ranges from (0, 0, 0) to (255, 255, 255), rendered in both Gamma mode and Linear mode. Here there is no difference between the two modes, unlike in panels (a) and (b). This shows that the Linear mode does not consistently apply a nonlinearity, which makes it difficult to render images in this mode that are even approximately guided by physical realism.

Measuring luminance

The focus of this article is on using Unity to calibrate luminance, but if this information is to be useful, then we also need a way of measuring luminances in a VR headset display. In this section, we briefly describe three such methods.

VR photometer. Specialized photometers and colorimeters are available for VR headsets. These devices produce calibrated measurements of the stimulus an observer sees when wearing a headset, and these measurements can be used with the color grading methods described below to linearize the relationship between achromatic RGB value and luminance. However, these devices are expensive, and many labs that run VR experiments do not have one, so we consider other approaches to measuring luminance as well.

Spot photometer. We used a Konica-Minolta LS-110 spot photometer to measure display luminance in an Oculus Rift S headset (single fast-switch LCD display, 2560 × 1440 resolution, 80 Hz refresh rate). We found that when the photometer's front lens was placed a few centimeters from the headset's eyepiece, after some positional adjustments, the view through the photometer showed a small patch of the LCD display. There was substantial blur, but images displayed in the headset were easily seen, and individual pixel boundaries were visible. It is important to keep in mind that this arrangement combines two devices with proprietary optical properties that were not designed to be interfaced with one another. Nevertheless, in Appendix A, we show that it is possible to use this configuration to make luminance measurements for calibration. We show that the main requirement for this approach is simply to display a very large calibration patch in the headset when measuring luminance, in order to minimize the effect of light scatter.

Pinhole camera. In the following section ("Color grading"), we describe an experiment where we used a pinhole camera to validate the luminance measurements made from a VR headset with a spot photometer. It is also possible to use such a pinhole camera to make the luminance measurements for calibration directly, bypassing the use of a spot photometer.

This is a simple and effective approach that provides calibration data without relying on assumptions about proprietary imaging systems. Appendix B describes an easy-to-build pinhole camera that can be used for this purpose.

Color grading

The Post-Processing Stack is an optional Unity package that provides several tools for modifying rendered images before display, including simulations of optical effects such as chromatic aberration (Post-Processing Stack v2, n.d.). It provides a method called "color grading," whose purpose is to adjust the color appearance of a rendered scene. Here we show that color grading can be used to compensate for a VR headset's nonlinear luminance response and make luminance proportional to achromatic RGB value.

Suppose we choose three $256 \times 256 \times 256$ arrays of integers from 0 to 255. We call them R, G, and B and denote their entries as R_{ijk}, B_{ijk}, and G_{ijk}, in the usual matrix- and tensor-like fashion, except that subscript indices begin at zero instead of 1. We can use these arrays to define a function M that transforms RGB triplets by mapping (r, g, b) to $M(r, g, b) \equiv (R_{rgb}, G_{rgb}, B_{rgb})$. For example, the RGB value (50, 100, 150) is mapped to $M(50, 100, 150) = (R_{50,100,150}, G_{50,100,150}, B_{50,100,150})$.

Color grading in Unity uses such a mapping, with two modifications. First, the maximum array size is $32 \times 32 \times 32$, and the full range of RGB values (0–255) is mapped using interpolation. Second, the arrays R, G, and B are encoded as a single $n \times n^2 \times 3$ color image. To construct this image, we rearrange the $n \times n \times n$ array R described above into an $n \times n^2$ array R' as follows. Let $R_{::k}$ denote the $n \times n$ subarray of R obtained by holding the third subscript constant at some value k. Then we define

$$R' = [R_{::0} \quad R_{::1} \quad \ldots \quad R_{::n-1}] \quad (1)$$

That is, the $n \times n$ slices of R obtained by setting $k = 0$, $k = 1$, and so on are concatenated left to right. More concisely,

$$R'_{i,j} = R_{i,(j \bmod n),\lfloor j/n \rfloor} \quad (2)$$

for $0 \leq i < n$ and $0 \leq j < n^2 - 1$, where "mod" is modulus and $\lfloor j/n \rfloor$ is integer division. Arrays G and B are rearranged into G' and B' in the same way. Finally, we construct an 8-bit color image with R', G', and B' as the three color channels and save the image in a lossless format such as PNG. In the Unity documentation, such an image is called a "lookup table (LUT) texture." When the LUT texture is loaded into the Post-Processing Volume component, the LUT transformation described above is applied as a late stage of calculating rendered RGB values. In the Supplementary Material, we provide MATLAB code for generating LUT texture images and instructions for incorporating the Post-Processing Stack into a Unity project (Supplementary Material, Section 1(f)).

Figure 3a shows that the color grading methods described here succeed in linearizing luminance on an Oculus Rift S headset. The red symbols show luminance measurements (made with an LS-110 spot photometer) from a set of achromatic RGB values assigned to a planar object with an "Unlit" material type and displayed in the headset. The plane was large enough to fill the headset's field of view, as recommended in Appendix A. The luminance values follow a typical expansive nonlinearity. The green symbols show the corresponding measurements after a LUT texture generated by code provided in the Supplementary Material was loaded into the Post-Processing Volume component. With the LUT in place, luminance is proportional to achromatic RGB.

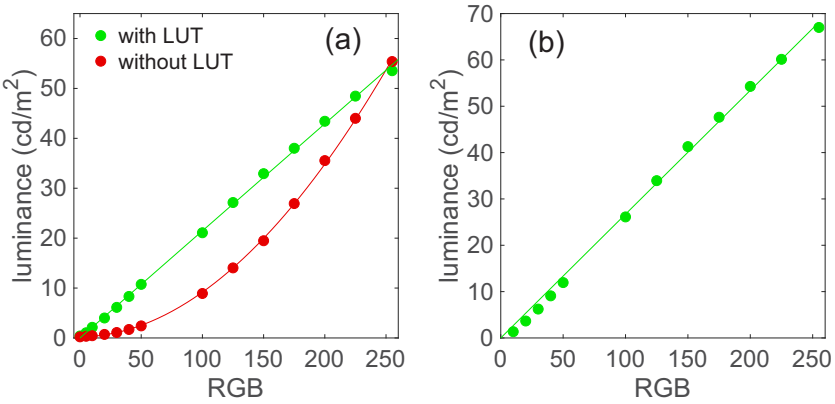

Figure 3. (a) Luminance versus achromatic RGB, with and without a linearizing LUT texture. The fit to the data without a LUT is a gamma function. The fit to the data with a LUT is a straight line constrained to pass through the origin. (b) Linearization confirmed with measurements from a pinhole camera.

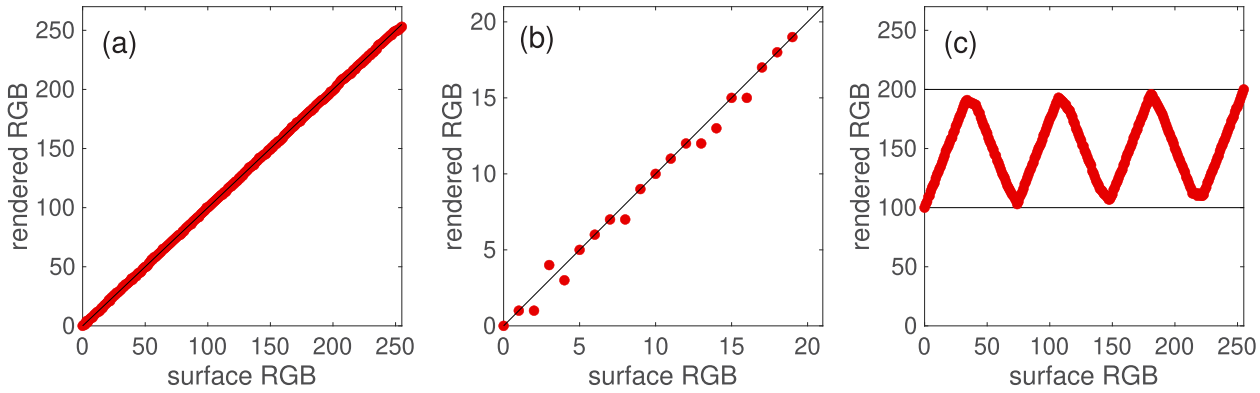

Figure 4. Rendered RGB value of an achromatic Lambertian surface as a function of its assigned RGB color, in Gamma mode, with color grading. (a) With a LUT texture that represents the identity mapping. (b) Same data as in panel (a), but showing a smaller range to highlight small departures from the identity mapping. (c) With a LUT texture that represents a triangular wave oscillating between RGB values of 100 and 200; note that the rendered RGB value does not usually reach 100 or 200.

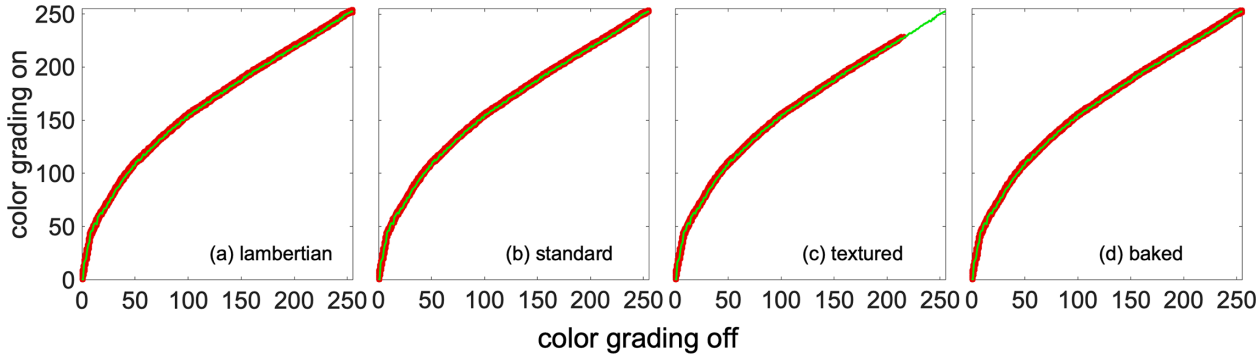

Figure 5. Effect of color grading on rendered achromatic RGB with various Unity settings. (a) Lambertian material. (b) Standard material. (c) Standard material with achromatic texture. (d) Standard material with baked lighting. In each panel, the red curve shows the RGB values of individual pixels rendered with and without color grading. The green curve shows the data from panel (a), superimposed on each red curve, to demonstrate that the four curves are identical.

As a further test of whether we had succeeded in linearizing luminance, Figure 3b shows luminance measurements made with the LUT in place, using the pinhole camera described in Appendix B, which does not rely on the model of light scatter used in Appendix A. Here too, luminance is proportional to achromatic RGB. We do note a slight dip in measured luminance below the linear fit at low luminances and a slight increase above at high luminances. This may be due to small calibration errors in the pinhole camera.

The Unity documentation reports that color grading interpolates RGB values linearly between the values specified in the LUT texture. We find that there are sometimes small departures from this claim. For example, Figures 4a and 4b show the result of repeating the experiment of Figure 2c but with color grading based on a LUT texture that represents the identity mapping. Here we expect rendered RGB to be equal to the Lambertian material's RGB value. This is approximately true, but the zoomed-in plot in Figure 4b shows that there are small and consistent deviations. Figure 4c shows a more demanding version of this test, with a LUT texture that has just eight samples and represents a triangular wave that oscillates between RGB values of 100 and 200. Rendered RGB follows the expected pattern approximately, but again with some deviations. Except for unusual mappings like the triangular wave, departures from linear interpolation usually seem to be limited to just one RGB value, so in most applications, they will not interfere with using color grading to linearize luminance. We mention these approximations, though, as another example of the need to examine rendered stimuli closely.

We wished to confirm that the effect of color grading was the same in a few other common Unity configurations. We rendered a sphere with (a) a

Lambertian material (Legacy/Diffuse), (b) a Standard material with default settings, (c) a Standard material with an achromatic texture, and (d) a Standard material with baked lighting. We rendered each sphere with and without color grading, using a LUT texture generated by the software in the Supplementary Material to linearize luminance in the Oculus Rift S headset used throughout this article. Figure 5 shows how color grading transformed the images, by plotting the achromatic RGB value rendered for each pixel with color grading against the achromatic RGB value of the same pixel without color grading. The effect of color grading was identical in all four cases.

Color grading in Unity differs from some other lookup table methods that readers may be familiar with, such as the one implemented for MATLAB in the Psychophysics Toolbox (PTB; Brainard, 1997). One difference is that the PTB lookup table is a 256 $\times$ 3 matrix (or $n \times 3$ for greater color depths), which we can call L, and an RGB triplet (r, g, b) is mapped to (L_{r1}, L_{g2}, L_{b3}). This means that each component of the triplet is transformed independently of the other components. In Unity color grading, the mapping of each component can depend on the values of the other components as well (recall that each component is effectively looked up in an $n \times n \times n$ array), so the transformation is much more flexible. With this degree of flexibility, it should be possible to use color grading to implement standard methods for color calibration as well as luminance calibration (Brainard et al., 2002).

Another difference is that the PTB lookup table does not affect the RGB values in video memory. Instead, the lookup table is written to graphics hardware, and RGB values in video memory are transformed en route to being displayed on the monitor. The RGB values in a screen capture, for example, are unaffected by the PTB lookup table. In Unity, the LUT texture is used in a late stage of the rendering process, and it does modify the RGB values written to video memory. One consequence is that if a display has a PTB-type lookup table, then it still needs to be controlled by some mechanism other than Unity color grading.

Discussion

VR is a promising tool that opens the way to creative new approaches to perceptual experiments. However, Unity and VR headsets are two complex, not always transparent, interfacing systems that were not primarily designed for research. As a result, it is important to take a cautious approach to stimulus generation. Which stimulus features call for the most attention will differ from one research area to another. Our research with VR has centered on lightness constancy, so we have examined the luminance properties of stimuli. In the work reported here, we have done this only for a subset of the many components available in Unity (materials, lighting types, render settings, etc.), so we still recommend a test-as-you-go approach. One feature that makes such experiments easier is that however complex the rendering process may be, Unity provides access to the pixel-by-pixel RGB values of the rendered images (see `MainScript.cs` in the Supplementary Material for a code-based method of taking screen captures in Unity), so with a few simple tests, it is usually possible to tell whether the stimulus is what we expect it to be.

There are other tools that are useful for developing VR experiments, which we have not explored here. As we have mentioned, specialized equipment is available for measuring luminance, color, and optical properties of headset displays. Another tool is the C# source code for Unity itself, which is available for reference (https://github.com/Unity-Technologies/UnityCsReference, https://github.com/Unity-Technologies/PostProcessing). This is a potentially valuable resource, but it is a large, complex, and evolving code base. It seems unlikely that many perception labs will thoroughly master this code, so experimental tests of rendered stimuli will continue to be important.

We have focused on software tools for luminance calibration in Unity. We have not discussed possible hardware limitations, which are also important for some applications. These limitations include pixel independence, color channel constancy, color channel additivity, and spatiotemporal homogeneity. These issues are not specific to VR displays and are discussed at length by Brainard et al. (2002). Color channel constancy and additivity are investigated for Unreal Engine and the HTC Vive by Toscani et al. (2019). The methods we have described here provide sufficient control over displayed RGB values to also make these tests straightforward in Unity.

We have discussed Unity's Built-in Render Pipeline, which is the default pipeline. Unity has alternative pipelines available as well. The Scriptable Render Pipeline allows users to program their own rendering routines, such as custom shaders. The Universal Render Pipeline is designed to render images on a wide range of platforms, and the High Definition Render Pipeline aims to provide greater physical realism. These alternatives are also worth investigating for VR experiments (e.g., Diaz-Barrancas et al., 2021), as they have the potential to provide greater transparency and flexibility and to allow users to create more realistic and precisely controlled virtual environments.

Keywords: virtual reality, methods, luminance, color, calibration

Acknowledgments

The authors thank Matthew Cutone, Vijay Singh, and Laurie Wilcox for helpful comments. This work was funded in part by an NSERC Discovery Grant to RFM and a VISTA Doctoral Scholarship to KYP.

Supplementary Material is available at https://osf.io/ryte5.

Commercial relationships: none.
Corresponding author: Richard, F. Murray.
Email: rfm@yorku.ca.
Address: Department of Psychology and Centre for Vision Research, York University, Toronto, Ontario, Canada.

Footnotes

[1]The default Unity material, called "Standard," is not Lambertian with its initial settings. To create a Lambertian surface, a material of type "Legacy Shaders/Diffuse" can be assigned to an object.
[2]The sRGB nonlinearity consists of a small linear segment joined to a nonlinear curve, and together they are closely approximated by $f(x) \sim x^{2.2}$.

References

Brainard, D. H. (1997). The psychophysics toolbox. *Spatial Vision, 10*, 433–436.

Brainard, D. H., Pelli, D. G., & Robson, T. (2002). Display characterization. In J. Hornak (Ed.), *Encyclopedia of imaging science and technology* (pp. 172–188). Hoboken, NJ: John Wiley & Sons, Inc.

Diaz-Barrancas, F., Cwierz, H., & Pardo, P. J. (2021). Real-time application of computer graphics improvement techniques using hyperspectral textures in a virtual reality system. *Electronics, 10*(22):2852, 1–10.

Greengard, S. (2019). *Virtual reality*. Cambridge, MA: MIT Press.

Pharr, M., Wenzel, J., & Humphreys, G. (2017). *Physically based rendering* (3rd ed.). New York: Morgan Kaufmann Publishers.

Post-Processing Stack v2. (n.d.). https://docs.unity3d.com/Packages/com.unity.postprocessing@3.2/manual/index.html.

Rodriguez, R. G., Bayer, F., Toscani, M., Guarnera, D., Guarnera, G. C., & Gegenfurtner, K. R. (2022). Colour calibration of a head mounted display for colour vision research using virtual reality. *SN Computer Science, 3*(22), 1–10.

Toscani, M., Gil, R., Guarnera, D., Guarnera, G. C., Kalouaz, A., & Gegenfurtner, K. R. (2019). Assessment of OLED head mounted display for vision research with virtual reality. In K. Yetongnon, A. Dipanda, G. Sanniti di Baja, L. Gallo, & R. Chebir (Eds.), *15th Annual Conference on Signal-Image Technology and Internet-Based Systems (SITIS)* (pp. 738–745). Los Alamitos, CA: IEEE Computer Society.

Unity Technologies. (2020). *Unity, version 2020.1.5f1*. Retrieved from https://unity.com

Unity User Manual 2020.1. (2020). https://docs.unity3d.com/2020.1/Documentation/Manual/UnityManual.html.

Appendix A. Measurements with a spot photometer

We noted in the main text that there appears to be substantial light scatter when viewing the Oculus Rift S display through a Konica-Minolta LS-110 photometer. To confirm this impression, we measured the luminance of a series of achromatic disks with a fixed luminance and a range of diameters, displayed on a black background, shown on the Rift S headset and on a flat-panel LCD monitor (2017 iMac, 27 in.). The nominal integration region of the LS-110 has a diameter of 0.33 degrees of visual angle, and we aimed this region at the center of each disk. Figure 6 shows that on the flat-panel monitor, additional stimulus area outside the integration region had little effect on measured luminance. On the VR headset, however, stimulus areas well outside the nominal integration region contributed to luminance measurements.

This raises the question of whether it is valid to use this apparatus to measure luminance on a VR display. In this appendix, we show that so long as we use a large calibration stimulus, this light scatter is not problematic for luminance calibration.

An idealized photometer reading p^* from a luminance image $I(\mathbf{x})$ on the headset display is the average luminance in a small target region R_T:

$$p^* = \frac{1}{A(R_T)} \int_{R_T} I(\mathbf{x})\, d\mathbf{x} \quad (3)$$

Here, $A(R_T)$ is the area of region R_T.

In fact, we find that the photometer we used (Konica-Minolta LS-110) captures significant stray light from within the headset (Oculus Rift S), as can be confirmed by noting that high-luminance patches outside the target region affect the luminance reading (Figure 6). The optics of the headset and photometer are presumably linear, so a more realistic and general model of the photometer reading p is a weighted

integral of the luminance image over the entire display region R:

$$p = \int_R w(\mathbf{x}) I(\mathbf{x}) d\mathbf{x} \quad (4)$$

The weighting function $w(\mathbf{x})$ varies with the position and orientation of the photometer.

During calibration, we might show an image $T(\mathbf{x})f(u)$ that just fills the ideal photometer's sensitive region R_T. Here, $T(\mathbf{x})$ is a spatial luminance pattern, such as a small disk, and the scaling function $f(u)$ represents the fact that we can control the luminance of this pattern with a parameter u. For example, $T(\mathbf{x})f(u)$ could be a small disk rendered with RGB triplet (u, u, u). We assume that $f(0) = 0$, and because the display is not completely dark even at RGB value (0, 0, 0), we represent the calibration stimulus as $I(\mathbf{x}) = T(\mathbf{x})f(u) + B(\mathbf{x})$, where $B(\mathbf{x})$ is the residual background image corresponding to $u = 0$. In general, $B(\mathbf{x})$ extends over the full display region R. The ideal photometer's response to this stimulus $I(\mathbf{x})$ is

$$p^* = \frac{f(u)}{A(R_T)} \int_{R_T} T(\mathbf{x})\, d\mathbf{x} + \frac{1}{A(R_T)} \int_{R_T} B(\mathbf{x})\, d\mathbf{x} \quad (5)$$

whereas the response of a photometer with light scatter is

$$p = f(u) \int_{R_T} w(\mathbf{x}) T(\mathbf{x})\, d\mathbf{x} + \int_R w(\mathbf{x}) B(\mathbf{x})\, d\mathbf{x} \quad (6)$$

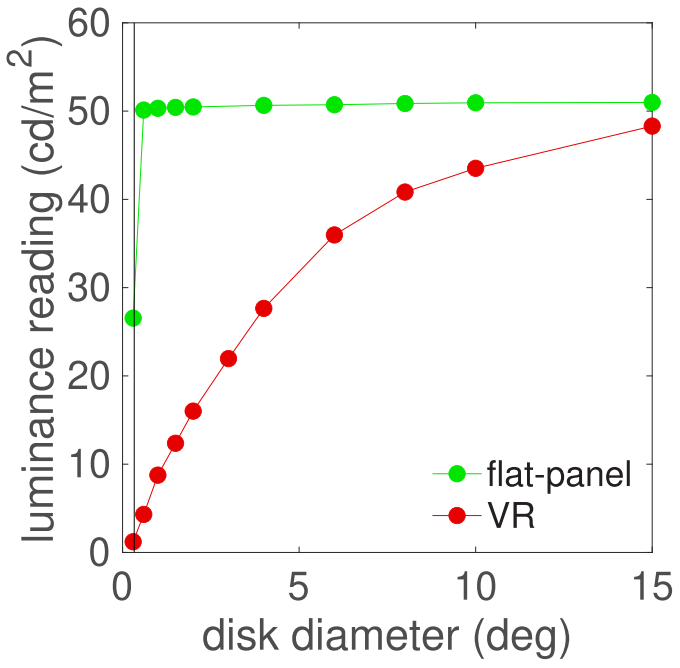

Figure 6. Luminance measurements from a photometer directed at the center of disks with a range of diameters and fixed luminance. The black vertical line at 0.33° indicates the diameter of the nominal integration region of the photometer.

If (a) $T(\mathbf{x})$ and $B(\mathbf{x})$ are constant within region R_T, and if (b) the integrand in the second integral in Equation 4 is nonzero only over the target region R_T, then these two readings are proportional to each other as a function of u, and we can use p as a proxy for p^* that simply loses an overall scale factor. We can meet condition (a), as the target region R_T is small, so stimuli with constant RGB values have approximately constant luminance over this area. Meeting condition (b), however, is more difficult: Due to scattering, light from a large area of the background image contributes to the photometer's response, so an unknown constant is added to all luminance readings relative to the ideal photometer's response.

A natural solution is to make the image $T(\mathbf{x})$ cover the entire display, just like the background $B(\mathbf{x})$, so that a large area of $T(\mathbf{x})$ contributes to the measurement as well. Under certain conditions, this approach succeeds. We cannot usually show a calibration image $T(\mathbf{x}) = tf(u)$ that is constant over the whole display, as most displays are brighter at the center than in the periphery. Instead, we assume that the stimulus $I(\mathbf{x})$ is the ideal calibration stimulus $T'(\mathbf{x}) = tf(u)$ and ideal background image $B'(\mathbf{x}) = b$ (both of which are independent of $\mathbf{x}$), spatially modulated by some function $g(\mathbf{x})$:

$$I(\mathbf{x}) = g(\mathbf{x})\big(\, tf(u) + b \,\big) \quad (7)$$

The responses of the idealized and light-scattering photometers to this stimulus are

$$p^* = \frac{tf(u) + b}{A(R_T)} \int_{R_T} g(\mathbf{x})\, d\mathbf{x} \quad (8)$$

$$p = \big(\, tf(u) + b \,\big) \int_R g(\mathbf{x}) w(\mathbf{x})\, d\mathbf{x} \quad (9)$$

These responses are proportional as a function of u, so if the assumptions supporting this approach are met, we can use a spot photometer to measure luminance in a headset up to a scale factor, even with substantial light scatter.

The main assumption in this approach is that the scaling function $f(u)$ is the same across the display. (This allows us to move $f(u)$ outside the integrals in Equations 8 and 9.) This is a reasonable assumption, and it underlies the calibration procedures used in practically all psychophysical experiments with flat-panel monitors, where it is assumed that a single pointwise transformation can linearize the relationship between RGB and luminance across the display. Nevertheless, we tested this assumption by measuring the luminance of a test patch as a function of u at several locations on the headset display. To make these measurements without relying on our model of light scatter, we used a custom-built pinhole camera. We describe the construction and calibration of the camera in Appendix B. We used this apparatus to measure the

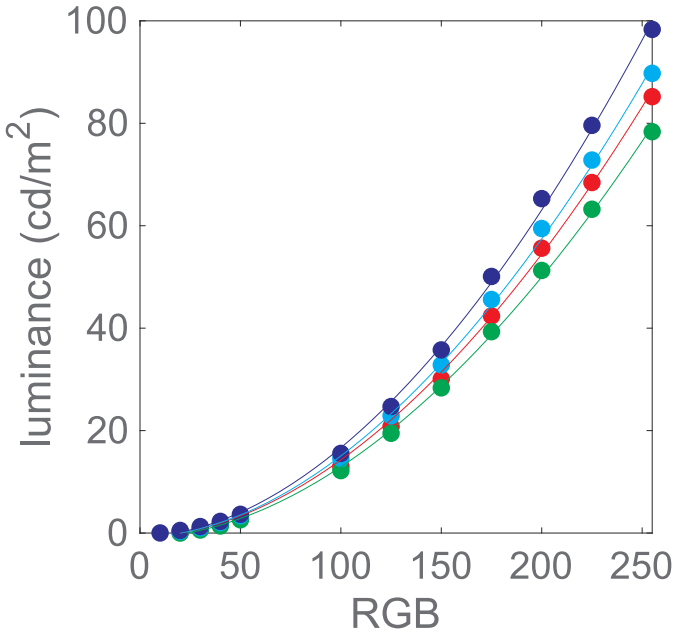

Figure 7. Luminance measurements at several locations across the central 5 to 10 degrees of the VR headset display. Luminance is shown as a function of achromatic RGB value, fitted with power functions constrained to have a common exponent.

luminance displayed as a function of achromatic RGB value, at four locations separated by 5 to 10 degrees of visual angle, in the central region of the headset display. Figure 7 shows that luminances at different locations were fit well by power functions $f(x) = kx^{\gamma} + \delta$ sharing a common exponent γ, consistent with the assumption being tested here.

Another assumption of this method is that the same function $g(\mathbf{x})$ spatially modulates the test and background images. (This allows the terms $tf(u)$ and b to be grouped together in Equations 8 and 9.) This assumption is more difficult to test without specialized equipment, but it is also less important if the photometer's response to the background image $B(\mathbf{x})$ is much smaller than its response to the calibration image $T(\mathbf{x})f(u)$. We find that when the calibration image covers the whole display, the photometer reading for RGB value (0, 0, 0) is 0.19 cd/m^2, and the reading for (255, 255, 255) is 59 cd/m^2. Thus, the background makes only a small contribution to luminance measurements from a full-field stimulus, and differences in the spatial modulation $g(\mathbf{x})$ of the foreground and background images should create only small departures from proportionality between ideal and real photometer readings.

A separate, more practical issue is that the photometer may capture stray light from the physical environment. We find that even when the photometer is aimed directly into the headset, turning the room lights on and off changes the luminance reading. To address this problem, we cover the photometer and headset with an opaque veil during calibration.

In the section "Color grading" in the main article, we show that calibration of a VR headset with measurements from a spot photometer can be validated using measurements from the pinhole camera described in Appendix B.

Appendix B. Measurements with a pinhole camera

Here we describe a pinhole camera apparatus that we constructed to measure luminance on a VR headset display, in order to validate the measurements we made with a spot photometer. We created pinhole pupils in the eyes of a life-sized mannequin head by drilling small holes (diameter ~1.0 mm) in discs of thin steel sheet metal (diameter 2 cm, thickness 0.002 in.), and mounting these disks on the outer side of larger holes drilled through the eyes of the mannequin. We painted the inside of the mannequin head matte black to increase opacity and reduce stray light. A clamp that we mounted inside the apparatus held a photodiode (model PIN-10-AP; OSI Optoelectronics) approximately 10 cm behind one of the pinholes. The spectral sensitivity of the photodiode closely matched the CIE photopic spectral luminous efficiency function. The active area of the photodiode measured 1 cm × 1 cm, so at a distance of 10 cm from the pupil, it integrated light over 5.7 × 5.7 degrees of visual angle. We used the photodiode in photovoltaic mode and passed its output through a current-to-voltage converter, a passive low-pass filter, a voltage amplifier, and finally the analog-to-digital converter of an Arduino UNO board. This apparatus provided a digital readout of the illuminance at the photodiode, in arbitrary units. We calibrated the readout by placing the apparatus (with the photodiode inside the mannequin head) in front of a calibrated flat-panel LCD monitor and measuring the mapping from known luminances on the monitor to digital readings from the apparatus. This allowed us to calculate the inverse mapping and to convert digital readings to luminance measurements in cd/m^2. We confirmed that after this calibration procedure, we were able to use the apparatus to measure known luminances displayed on the flat-panel monitor to within a few percent. We mounted the VR headset on the mannequin head and used the apparatus to measure luminance on the headset display. To measure luminance at different locations on the display, we manually adjusted the position of the photodiode inside the apparatus.