Real-Time 3D/XR Visualization

Practical Work 2: Shaders and Color Sciences

March 2026

1 Introduction

In this practical work you will build a complete Three.js application that uses **custom GLSL shaders** (targeting WebGL) to visualise color information extracted from an image or a video stream. You will explore six standard color spaces (RGB, HSV, CIEXYZ, CIExyY, CIELAB and CIELCH), generate **elevation maps** driven by individual color components, apply a **Lambertian directional lighting** model, and make every visualisation accessible in **VR and MR** through the WebXR API.

The reference images in Section 2 illustrate the expected visual language for the point-cloud and density-based visualisations you must implement and extend to all six target color spaces.

Note: You must create one self-contained web application per exercise. In **every** case, the same code base must support *desktop 3D visualisation*, *VR*, and *MR*.

2 Reference Examples

The following six images illustrate color-space visualisations in three of the target spaces. They show both *direct* point-cloud visualisation and *density-based* visualisation. Your work must at least match this visual quality, and you must extend it to cover all six color spaces listed in this document.

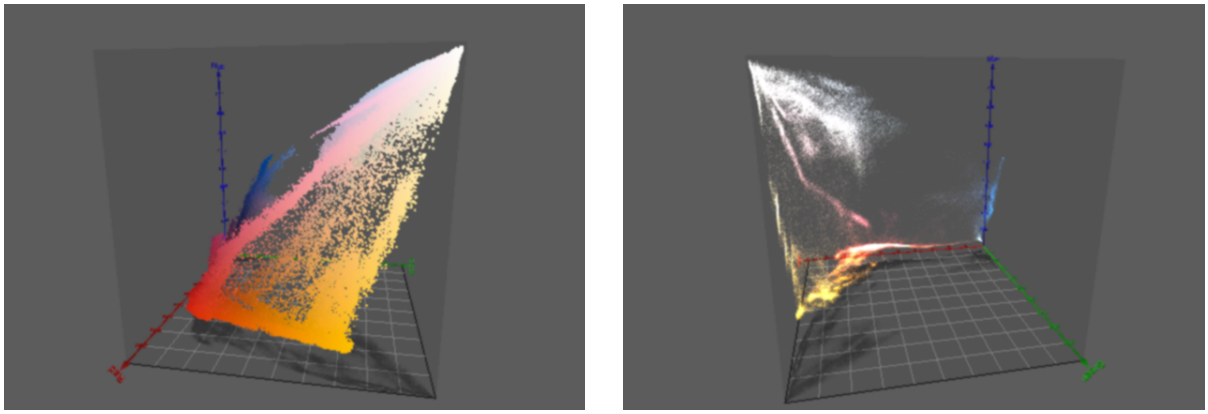

Figure 1: RGB color space — left: direct point-cloud visualisation; right: density-based visualisation.

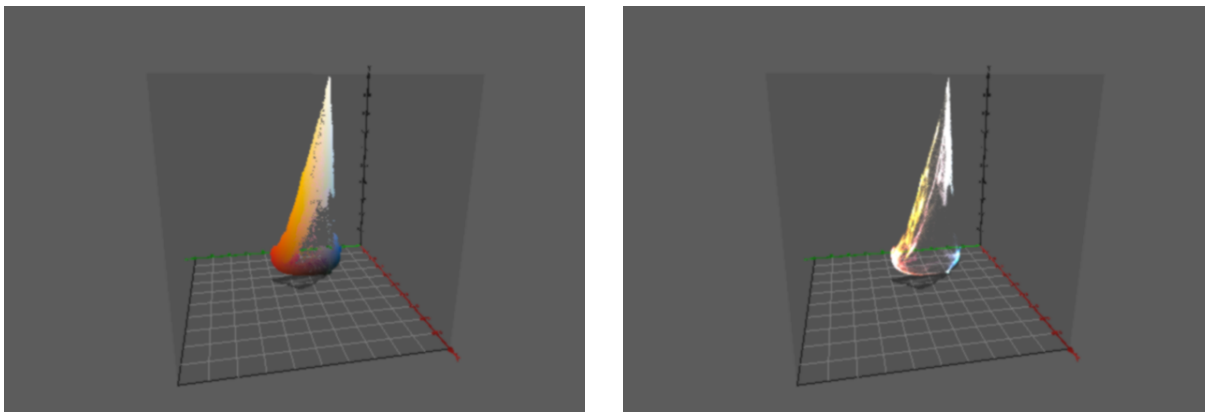

Figure 2: CIE xyY color space — left: direct point-cloud visualisation; right: density-based visualisation.

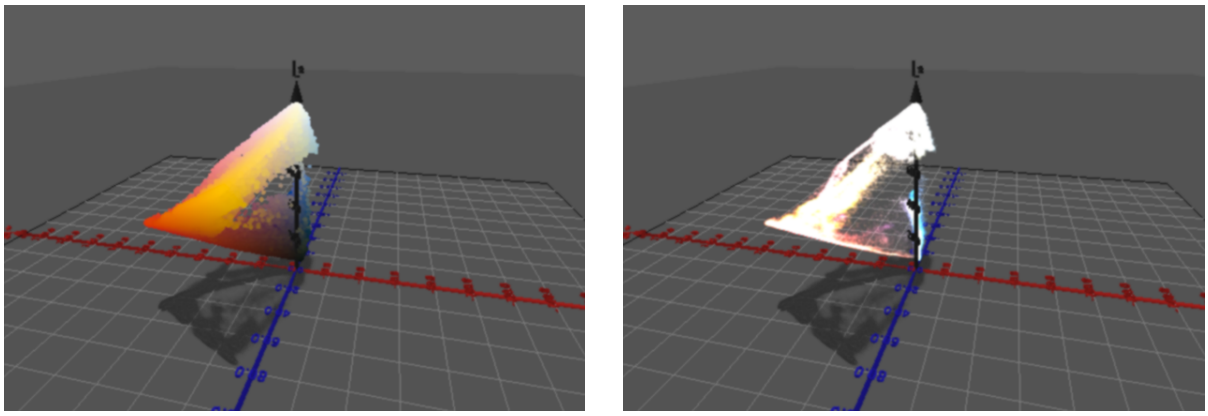

Figure 3: CIELAB color space — left: direct point-cloud visualisation; right: density-based visualisation.

3 color-Space Conversion Reference

All exercises rely on the same conversion pipeline. This section provides the formulae you must implement in your GLSL shaders. Unless stated otherwise, assume an **sRGB input signal** and use the **D65 reference white** $(X_n, Y_n, Z_n) = (0.95047,\ 1.00000,\ 1.08883)$.

3.1 sRGB to Linear RGB

For each sRGB component $C \in [0, 1]$:

$$C_{\text{lin}} = \begin{cases} \dfrac{C}{12.92}, & C \le 0.04045 \\ \left(\dfrac{C + 0.055}{1.055}\right)^{2.4}, & C > 0.04045 \end{cases}$$

3.2 Linear RGB → CIE XYZ

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.4124564 & 0.3575761 & 0.1804375 \\ 0.2126729 & 0.7151522 & 0.0721750 \\ 0.0193339 & 0.1191920 & 0.9503041 \end{bmatrix} \begin{bmatrix} R_{\text{lin}} \\ G_{\text{lin}} \\ B_{\text{lin}} \end{bmatrix}$$

3.3 CIE XYZ → CIE xyY

$$x = \frac{X}{X + Y + Z}, \qquad y = \frac{Y}{X + Y + Z}, \qquad Y = Y$$

with the convention $(x, y, Y) = (0, 0, 0)$ when $X + Y + Z = 0$.

3.4 CIE XYZ → CIELAB

Define the helper function

$$f(t) = \begin{cases} t^{1/3}, & t > \delta^3 \\ \dfrac{t}{3\,\delta^2} + \dfrac{4}{29}, & \text{otherwise} \end{cases} \qquad \text{where } \delta = \frac{6}{29}.$$

Then

$$L^* = 116\, f\left(\frac{Y}{Y_n}\right) - 16, \qquad a^* = 500\left[f\left(\frac{X}{X_n}\right) - f\left(\frac{Y}{Y_n}\right)\right], \qquad b^* = 200\left[f\left(\frac{Y}{Y_n}\right) - f\left(\frac{Z}{Z_n}\right)\right].$$

3.5 CIELAB → CIELCH

$$C^* = \sqrt{a^{*2} + b^{*2}}, \qquad h = \operatorname{atan2}(b^*,\, a^*)$$

The triple (L^*, C^*, h) forms a cylindrical representation of the CIELAB space.

3.6 RGB → HSV

Given $(R, G, B) \in [0, 1]^3$, let

$$C_{\max} = \max(R, G, B), \qquad C_{\min} = \min(R, G, B), \qquad \Delta = C_{\max} - C_{\min}.$$

Then:

$$V = C_{\max}$$

$$S = \begin{cases} 0, & C_{\max} = 0 \\ \dfrac{\Delta}{C_{\max}}, & \text{otherwise} \end{cases}$$

$$H = \begin{cases} 0^\circ, & \Delta = 0 \\ 60^\circ \times \dfrac{G - B}{\Delta} \mod 360^\circ, & C_{\max} = R \\ 60^\circ \times \left(\dfrac{B - R}{\Delta} + 2 \right), & C_{\max} = G \\ 60^\circ \times \left(\dfrac{R - G}{\Delta} + 4 \right), & C_{\max} = B \end{cases}$$

When used as a 3D coordinate, normalise H to $[0, 1]$ by dividing by 360.

4 Exercise 1: color-Space Point-Cloud Visualisation

Your first objective is to build a **shader-driven visualisation system** that displays color distributions extracted from an image or a video stream. The user must be able to switch interactively between the six target color spaces: RGB, HSV, CIEXYZ CIExyY, CIELAB and CIELCH.

5 Exercise 2: color Elevation Maps

In this exercise you will build **color elevation maps** (also known as *heightfield surfaces*). The principle is as follows: for every pixel (u, v) in the source image or video frame, compute one color component in a chosen color space and use its value as the height z. The resulting surface $z = h(u, v)$ provides a spatial visualisation of how a specific color channel varies across the image.

6 Exercise 3: Lambertian Directional Lighting on Elevation Maps

A flat color alone does not convey the 3D shape of the elevation surface. In this exercise you will add a **directional Lambertian lighting** model to improve depth perception on the color elevation maps built in Exercise 2.

6.1 Simplified Lambertian Model

We consider the surface of the elevation map to be a perfectly diffuse (*Lambertian*) reflector. For a single directional light source, the reflected intensity at a surface point is:

$$\boxed{I = I_d \cdot k_d \cdot \max(0,\ \mathbf{N} \cdot \mathbf{L})} \tag{1}$$

where:

- I_d is the **intensity** (color) of the directional light source.
- $k_d \in [0, 1]$ is the **diffuse reflectance** coefficient of the surface material.
- $\mathbf{N}$ is the **unit surface normal** at the shading point.
- $\mathbf{L}$ is the **unit direction vector** pointing *from the surface toward the light*.
- The $\max(0, \cdot)$ term prevents negative contributions (back-facing geometry receives no light).

6.2 Normal Estimation from the Heightfield

If the elevation surface is defined as $z = h(u, v)$, the unnormalised normal can be estimated from partial derivatives:

$$\mathbf{N} = \text{normalize}\left(\begin{bmatrix} -\dfrac{\partial h}{\partial u} \\ -\dfrac{\partial h}{\partial v} \\ 1 \end{bmatrix}\right) \tag{2}$$

In practice, use **finite differences** from the texture:

$$\frac{\partial h}{\partial u} \approx \frac{h(u+\epsilon,\ v) - h(u-\epsilon,\ v)}{2\,\epsilon}, \qquad \frac{\partial h}{\partial v} \approx \frac{h(u,\ v+\epsilon) - h(u,\ v-\epsilon)}{2\,\epsilon}$$

where ϵ is the texel size (i.e. $1/$grid resolution).

6.3 Ambient Term (Optional Extension)

To avoid completely black areas where $\mathbf{N} \cdot \mathbf{L} \leq 0$, you may add a constant ambient term:

$$I = I_a \cdot k_a + I_d \cdot k_d \cdot \max(0, \mathbf{N} \cdot \mathbf{L})$$

where I_a and k_a are the ambient light intensity and ambient reflectance, respectively.

Remarks on Full WebXR Integration (VR and MR)

These exercise focuses on making complete applications fully usable in VR and MR. The goal is not simply to render the scene in a headset, but to provide an **interactive, immersive workflow** where all parameters can be adjusted without leaving the XR session.

Reference: Three.js WebXR VR Sandbox

Study the source code of the following demo before starting these exercises:
https://threejs.org/examples/?q=vr#webxr_vr_sandbox

It demonstrates how to create interactive 3D UI panels.

Submission

Submit your source code (HTML, JS, assets, etc.) as a web pages (one per exercise) and a git repository.