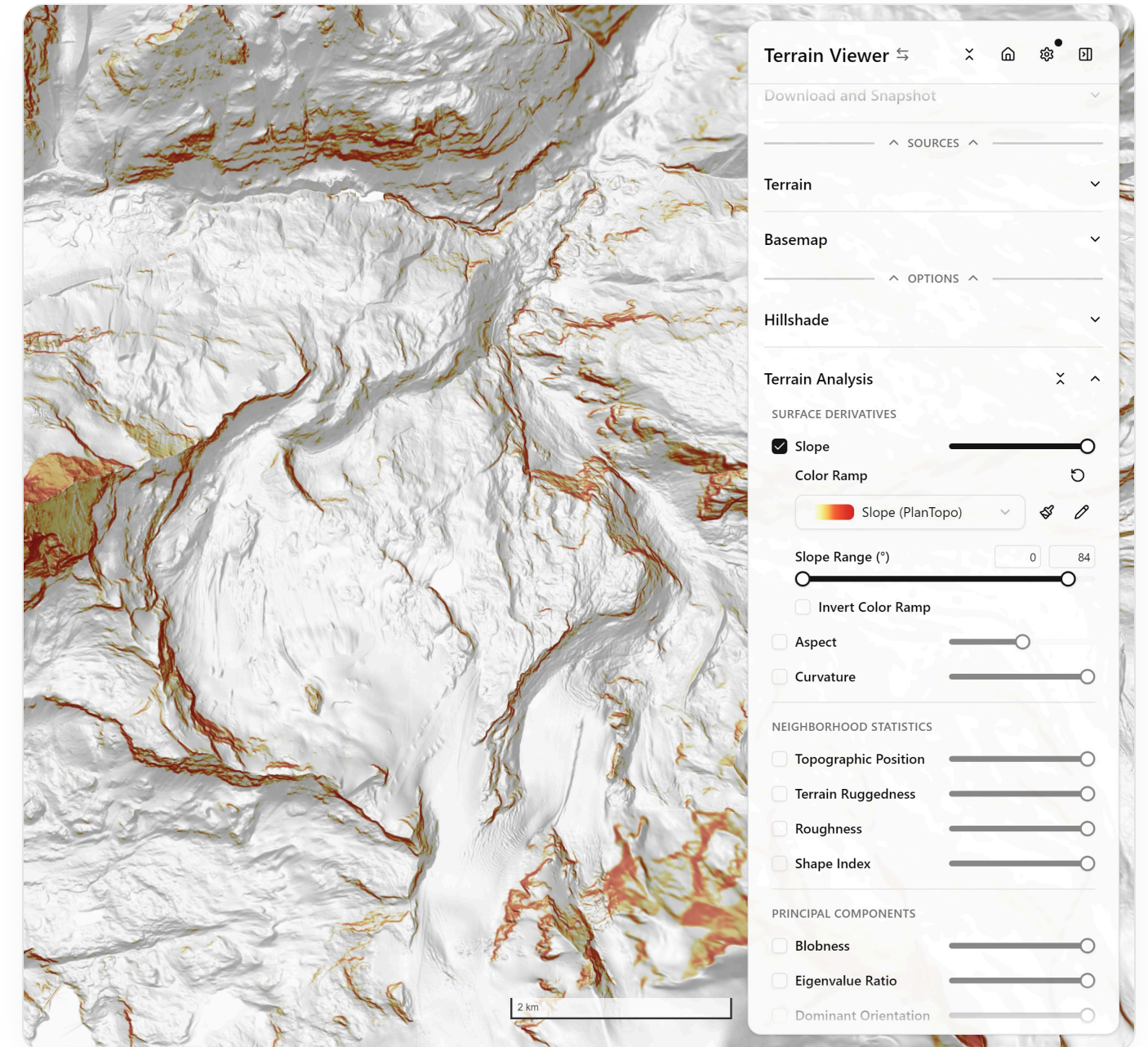
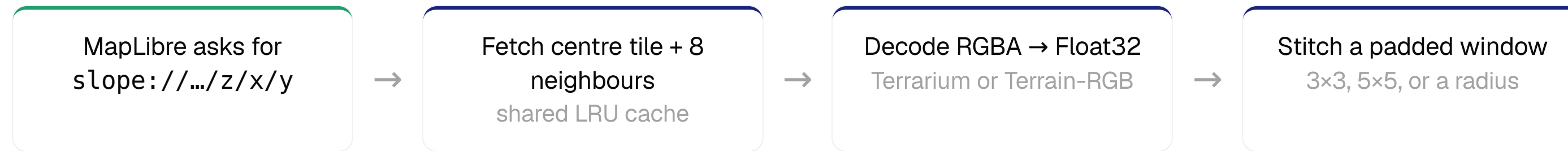


Protocols, kernels, caches

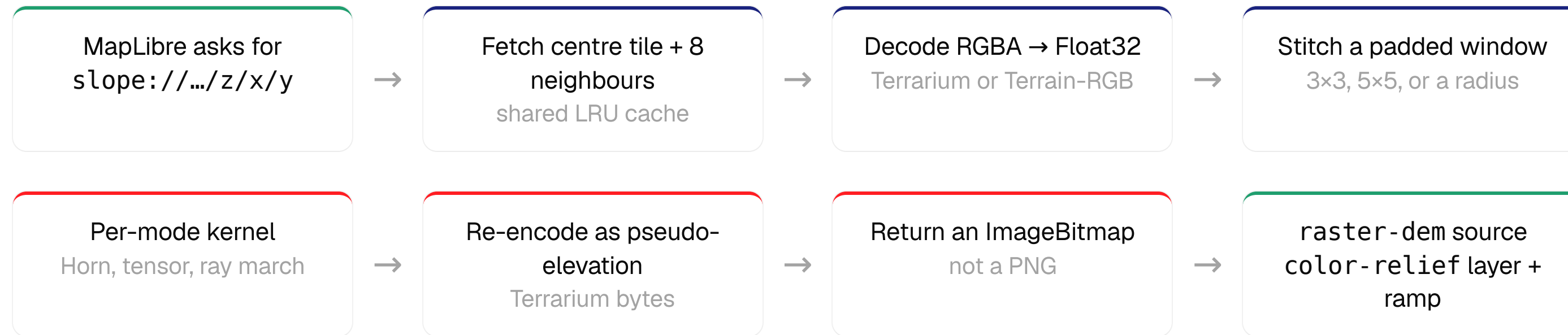
How a raster-dem tile becomes slope, sky-view factor or Phong shading without a server, and the one registry that holds it together.



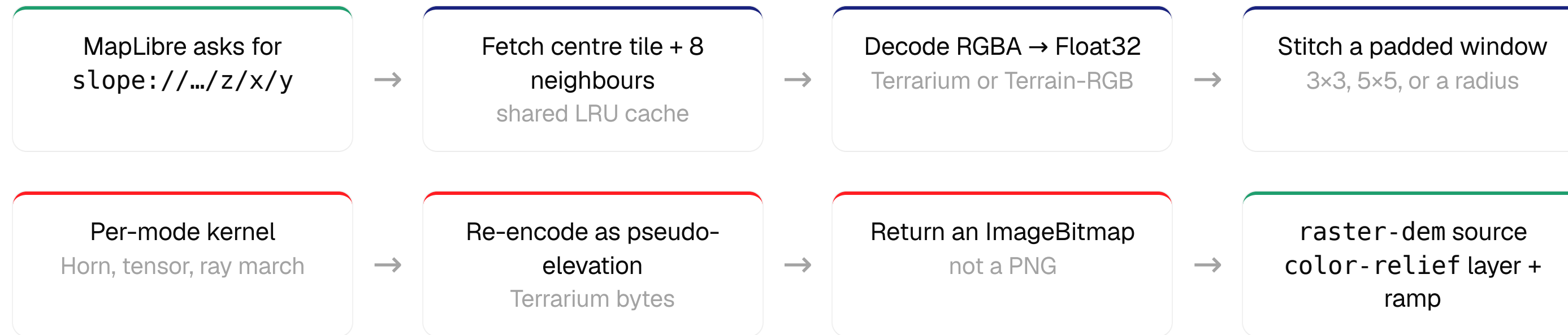
A custom protocol is a function MapLibre calls for a tile



A custom protocol is a function MapLibre calls for a tile



A custom protocol is a function MapLibre calls for a tile



■ The trick

The derivative is smuggled through the elevation pixel format. MapLibre's own color-relief decodes and colours it; no mode hand-rolls an RGBA colormap.

■ Eleven modes, one pipeline

slope, aspect, curvature, TRI, TPI, roughness, blobness, SVF, openness, local dominance... one formula each. LRM and lighting diverge.

Decoding, and the Horn 3×3 gradient

TERRARIUM

$$h = (R \cdot 256 + G + B/256) - 32768$$

MAPBOX TERRAIN-RGB

$$h = -10000 + (R \cdot 256^2 + G \cdot 256 + B) \cdot 0.1$$

HORN GRADIENT, WINDOW A0..A8 ROW-MAJOR, L = GROUND RESOLUTION

$$\begin{aligned}\partial z / \partial x &= (a_0 + 2a_3 + a_6 - a_2 - 2a_5 - a_8) / (8 \cdot L \cdot \cos \text{lat}) \\ \partial z / \partial y &= (a_6 + 2a_7 + a_8 - a_0 - 2a_1 - a_2) / (8 \cdot L \cdot \cos \text{lat})\end{aligned}$$

SLOPE

$$\text{atan}(\sqrt{(\partial z / \partial x)^2 + (\partial z / \partial y)^2}) \cdot 180 / \pi$$

ASPECT

$$\begin{aligned}\theta &= \text{atan2}(\partial z / \partial y, -\partial z / \partial x) \\ \text{aspect} &= (90 - \theta) \bmod 360\end{aligned}$$

TRI · TPI · ROUGHNESS

$$\sqrt{\sum (a_i - a_4)^2} \cdot a_4 - \text{mean}(a_{i \neq 4}) \cdot \max - \min$$

Ported from GDAL's GDALSlopeHornAlg, Mercator-corrected by $\cos(\text{lat})$ at the tile centre.

■ Curvature

One Zevenbergen & Thorne fit, five readings

SECOND-ORDER FIT (P, Q FIRST PARTIALS; R, T, S SECOND)

$$\begin{aligned} p &= (a5 - a3)/2L & r &= (a5 - 2a4 + a3)/L^2 \\ q &= (a7 - a1)/2L & t &= (a7 - 2a4 + a1)/L^2 \\ s &= (a2 - a0 - a8 + a6)/4L^2 & g &= p^2 + q^2 \end{aligned}$$

PROFILE (FLOW ACCELERATION) · PLAN (CONVERGENCE)

$$\begin{aligned} \text{profile} &= 100 \cdot (r \cdot p^2 + 2spq + t \cdot q^2) / (g \cdot (1+g)^{1.5}) \\ \text{plan} &= 100 \cdot (r \cdot q^2 - 2spq + t \cdot p^2) / g^{1.5} \end{aligned}$$

COMBINED · DET HESSIAN

$$\nabla^2 z = (a1 + a3 + a5 + a7 - 4a4)/L^2 \quad rt - s^2$$

PRINCIPAL CURVATURES

$$\kappa_{1,2} = (r + t \pm \sqrt{(r-t)^2 + 4s^2}) / 2$$

CASORATI · SHAPE INDEX

$$\begin{aligned} &\sqrt{(\kappa_1^2 + \kappa_2^2)/2} \\ &(2/\pi) \cdot \text{atan2}(\kappa_1 + \kappa_2, \kappa_1 - \kappa_2) \in [-1, 1] \end{aligned}$$

STRUCTURE TENSOR (BLOBNESS, 5×5 HALO)

$$\begin{aligned} J &= [I_{xx} \ I_{xy}; \ I_{xy} \ I_{yy}] \text{ of box-averaged } gx^2, gy^2, \\ &gx \cdot gy \\ \text{blobness} &= \det J / \text{tr } J \cdot 100/64 \\ \text{eigen ratio} &= \lambda_{\min}/\lambda_{\max} \end{aligned}$$

One horizon march, three aggregations

HORIZON ANGLE, 8 COMPASS DIRECTIONS, UP TO THE SEARCH RADIUS

$$\theta(\text{dir}) = \max \text{ over } r=1..R \text{ of } \text{atan2}(h(r) - h_0, r \cdot L)$$

SKY-VIEW FACTOR (ZAKŠEK 2011)

$$\text{SVF} = (1 - \text{mean}(\sin(\max(\theta_i, 0)))) \cdot 100$$

OPENNESS (YOKOYAMA 2002), NOT CLAMPED

$\text{openness} = \text{mean}(90^\circ - \theta_i)$
negative openness = the same on $-h$

LOCAL DOMINANCE (HESSE 2016), OBSERVER AT 1.6 M

$$\text{mean over dirs} \times \text{distances of } \text{atan}((z_0 + 1.6 - z(d)) / d)$$

Why SVF feels slower every tile needs the DEM well beyond its footprint; a view paints together once its neighbourhood decoded.

A max cannot be coarsened skipping pixels could miss the narrow obstruction that sets the horizon. A mean can.

So local dominance samples its far field from pyramid ancestors, one ring per octave.

Hard shadows the same march in one direction only, the sun's real azimuth: in shadow if $\theta > \text{altitude}$.

The pyramid is the low-pass filter

LOCAL RELIEF MODEL

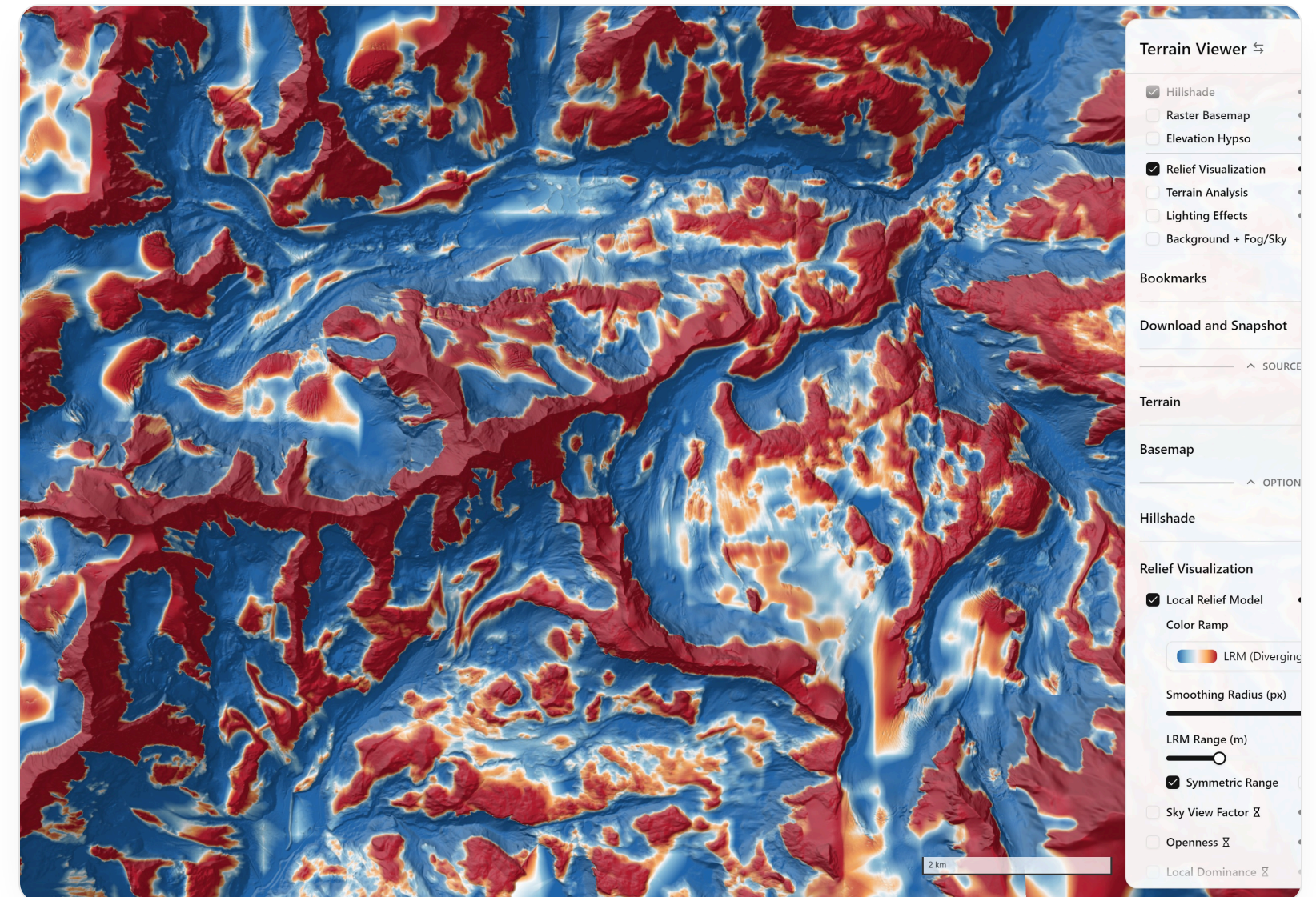
```
LRM = h_fine(native zoom) - bilinear(h_ancestor, k levels up)
```

```
export function radiusToLevels(radiusPx: number): number {
  return Math.min(6, Math.max(1,
    Math.round(Math.log2(Math.max(2, radiusPx)))))
}
// ancestorZ = z - k ; ancestorX = x >> k ; ancestorY = y >> k
const ancestorPxX = (xOffsetTiles * n + col + 0.5) / scale - 0.5
```

Free the tile server or COG overview already did the averaging; one ancestor backs up to 4^k fine tiles.

Half-pixel correction ancestor pixel i sits at $i + 0.5$; without it LRM correlated with slope at $-0.3 \dots -0.6$, with it ~ 0 .

WMS has no pyramid so the ancestor is requested small and upsampled client-side.



Radius 64 px: $k = 6$, a broad regional trend

■ Lighting effects

A real normal, then shading on the GPU

NORMAL FROM THE HORN GRADIENT

$$\ell = 1/\sqrt{dx^2 + dy^2 + 1} \quad n = (-dx \cdot \ell, -dy \cdot \ell, \ell)$$

PHONG: AMBIENT 0.35, SHININESS 32, $H = \text{NORMALIZE}(L + V)$, $V = (0,0,1)$

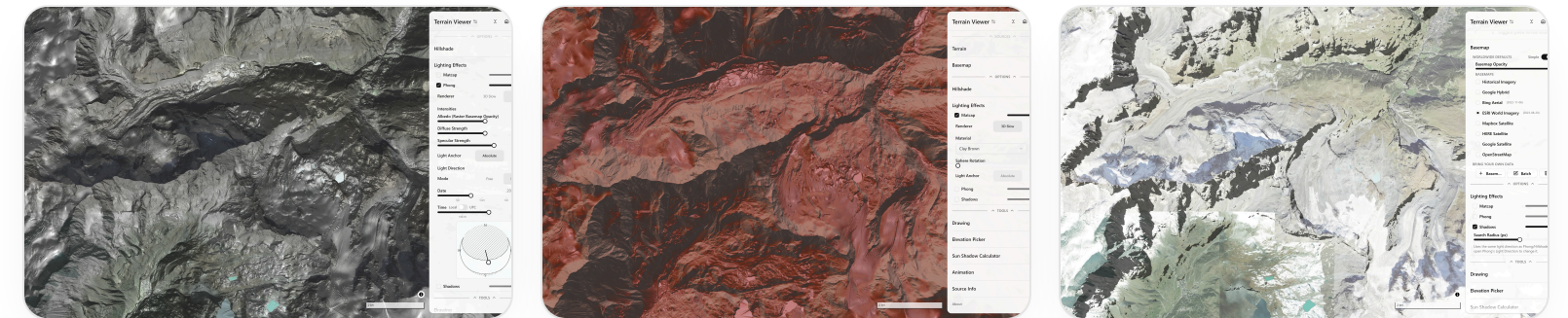
```
diffuse = kd · max(N·L, 0)
specular = ks · max(N·H, 0)^32
total = clamp(0.35 + diffuse, 0, 1) + specular
```

FRESNEL RIM (SCHLICK), LIVE RENDERER ONLY

$$\text{rim} = (1 - n \cdot v)^p$$

MATCAP

$$uv = \text{rotate}(nx, ny, \theta) \cdot 0.5 + 0.5 \rightarrow \text{texture}(uv)$$



Two paths a raster protocol MapLibre drapes (bakes a nadir view), and a live CustomLayerInterface where a slider is a uniform write.

View-dependent terms specular, camera matcap, fresnel: only the live path can draw them.

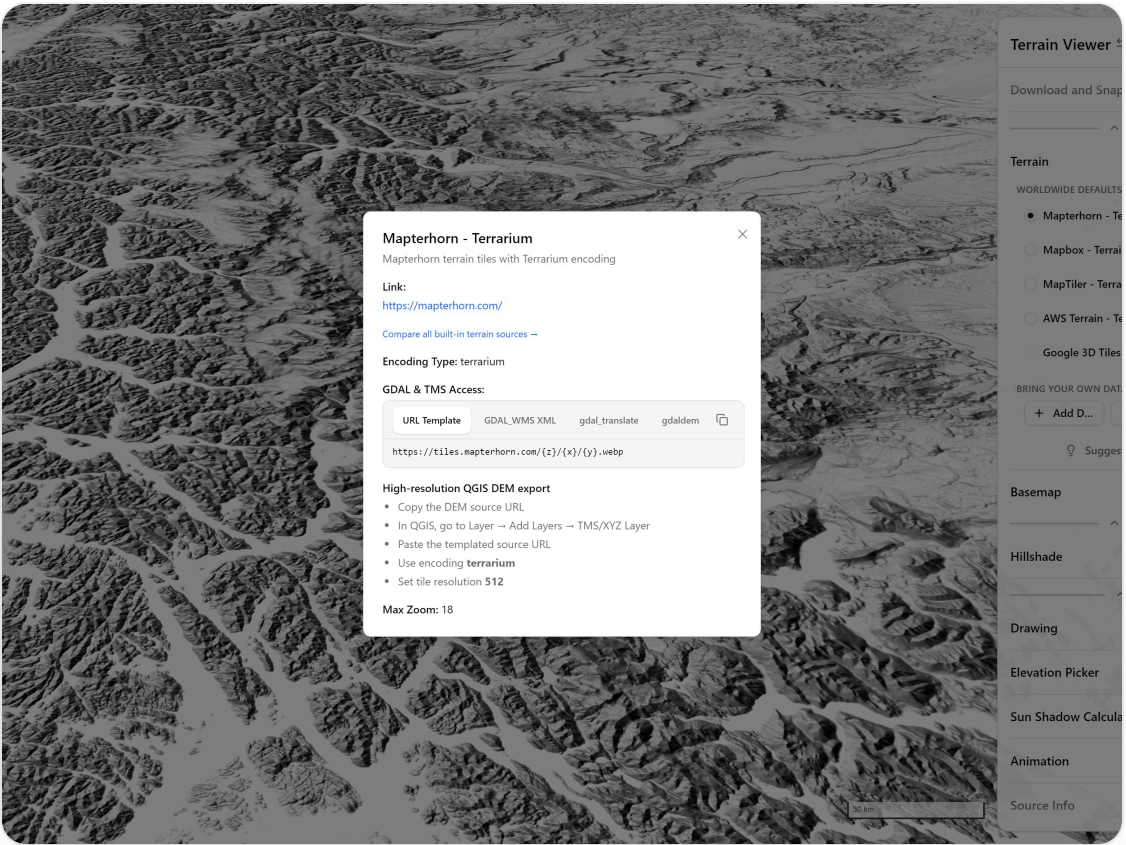
total ≤ 1 black at alpha 1 – total: multiply. total > 1 : white at total – 1: a highlight.

Three coloured lights diffuse averaged per channel: the duotone LiDAR look with real specular.

Elevation MapLibre cannot read, as Terrarium

<code>float32dem://</code>	a WMS or WCS returning float32 GeoTIFF per tile; WCS 2.0 via a subset=rewrite
<code>lerc://</code>	Esri LERC tiles from an ArcGIS ImageServer, decoded in the browser
<code>vrt://</code>	a GDAL VRT mosaic of COGs in any CRS, reprojected with proj4
<code>quantized-mesh://</code>	Cesium quantized-mesh terrain, rasterised
<code>demdiff://</code>	the difference of two sources: an nDSM, a change layer
<code>cog://</code>	geomatico: Cloud-Optimized GeoTIFF by range requests
<code>pmtiles://</code>	Protomaps archives
<code>dem-contour://</code>	maplibre-contour's own worker, fed through the registry

Region readers: a WMS, a VRT or a difference answers a whole export area in a few requests.



Source info: encoding, licence, GDAL export

Each mode is a scheme reading another scheme

<code>slope:// aspect://</code>	Horn gradient; slope packed Terrain-RGB, 0.1° step
<code>curvature:// tpi://</code> <code>tri:// roughness://</code>	3×3 window, Terrarium packing, curvature ×1000 on the wire
<code>blobness://</code>	structure tensor, 5×5 halo: blobness, eigen ratio, orientation
<code>svf:// openness:// local-</code> <code>dominance://</code>	ray-marched horizon, wrapped in withSlowTileStats
<code>lrm://</code>	ancestor-tile low-pass
<code>normals:// matcap://</code> <code>phong:// shadow://</code>	a real normal (WebGL2), shading, single-ray shadows
<code>tells:// isoband://</code> <code>unpaper://</code>	mound candidates and iso-line polygons as vector tiles; old maps without their paper
<code>cog-contour://</code>	contours from a COG in a worker

`slope://<upstream template>/{z}/{x}/{y}`
`// upstream may itself be`
`vrt://... or lerc://... or demdiff://...`

- **Key = URL**
buildProtocolUrl puts the upstream, the encoding, the tile size and every mode parameter in the URL. Change the sun azimuth: a different key. No invalidation logic anywhere.

One registry ties sources and modes together

```
import { registerProtocol, fetchTileBitmap, dispatchTile } from
"@/lib/protocol-registry"

// TerrainViewer.tsx, once at startup
registerProtocol("vrt", withTileResultCache(vrtProtocol))
registerProtocol("slope", withTileResultCache(slopeProtocol))
registerProtocol("svf",
withTileResultCache(withSlowTileStats("svf", svfProtocol)))

// anywhere: custom scheme → its handler, http(s) → fetch()
const bitmap = await fetchTileBitmap("vrt:///.../15/8074/14743", signal)
```

■ The rule

Register through registerProtocol, never maplibregl.addProtocol. Fetch a tile URL through fetchTileBitmap or dispatchTile, never fetch().

Why MapLibre dispatches custom schemes for its own sources only; fetch("slope:///...") fails.

Who else fetches every derived mode reading its upstream, the GeoTIFF export, the 2D picker and profile, the contours, the mound detector.

Twice missed LERC and quantized mesh, then VRT and demdiff: hillshade worked, every other mode drew nothing.

Now a new source type works in every mode, export, picker and contour the day it is registered.

On globalThis so a hot-reloaded copy of the module sees the same table.

Two caches, one ownership rule

■ sharedTileCache

Decoded upstream DEM tiles, float arrays + validity mask. Keyed on upstream URL + encoding. Saves the fetch and the decode: Slope after Hillshade downloads nothing.

■ tileResultCache

Finished protocol output, an ImageBitmap or MVT bytes. Keyed on the full protocol URL. 96 MB LRU. Saves the per-pixel work: toggling a mode off and on is instant.

■ Clone both ways

MapLibre transfers the ArrayBuffer to its worker and may close() a bitmap. Hand out the cache's own copy and the second hit throws DataCloneError. Clone before storing, clone before returning.

■ ImageBitmap, not PNG

convertToBlob(image/png) then MapLibre decodes it back: 99 ms median per tile.
createImageBitmap: 0.1 ms. maplibre-gl-js #8515 / #8525.

■ For scale

LRM over a z13 viewport: ~1.2 s of recompute.
quantized-mesh:// went from ~1000 ms to 3-97 ms per tile; lerc:// from ~1100 to 66-233.

```
const hit = lru.get(params.url)
if (hit) return { data: isBitmap(hit) ? await createImageBitmap(hit) : hit.slice() }
const result = await inner(params, abortController)
if (isBitmap(result.data)) put(params.url, await createImageBitmap(result.data))
```