TextureView

User Manual

Roman Meusch

June 2026

Contents

Chapter 1

Introduction

TextureView is a small, native macOS image and texture viewer built directly on Metal. It is aimed at graphics and engine developers who need to inspect the actual contents of a texture — not just a flat thumbnail. Alongside ordinary image formats it understands DDS textures with array slices, cube faces, mip chains, block-compressed pixel formats, and true high-dynamic-range data, and it lets you step through every one of those dimensions.

The app follows a deliberately simple rule: **one texture per window.** Open three files and you get three windows; close the last one and the app quits. There is no document browser, no tabs, and no inspector panel — every control sits in a compact frosted bar over the image itself.

1.1 Requirements

- macOS 12 (Monterey) or later.
- A Metal-capable Mac. Viewing block-compressed (BC) DDS textures additionally requires a GPU that supports BC texture compression; if yours does not, TextureView reports this when you try to open such a file rather than showing corrupt pixels.

Chapter 2

Opening images

There are four ways to open an image, and they all behave the same way once the file is loaded — a new window appears showing that texture.

- **File > Open...** (`Cmd-O`). Choose one or more files; each opens in its own window.
- **Finder.** Double-click a file that TextureView handles, or use *Open With > TextureView* from the Finder context menu.
- **Drag and drop.** Drag one or more image files from Finder onto any open TextureView window. Each dropped file opens in a new window.
- **Command line.** Open a file with the app from a terminal:

  ```
  open -a /Applications/TextureView.app /path/to/image.png
  ```

If you launch TextureView with nothing to show, it immediately presents an Open dialog; if you cancel that dialog, the app quits, since there is nothing to display. Likewise, **closing the last open window quits the app.**

If a file cannot be read or is in an unsupported format, TextureView shows a short dialog explaining why (for example, an unsupported DDS pixel format, or a BC-compressed texture on a GPU that cannot display it) instead of failing silently.

2.1 Supported input formats

2.1.1 Image files

Standard images are shown as 8-bit sRGB; Radiance `.hdr` is kept as linear floating point and displayed in HDR. The supported formats, and the practical limits of each, are:

Format	Notes
PNG	1/2/4/8/16-bit per channel
JPEG	Baseline and progressive (not 12-bit or arithmetic-coded)
BMP	Uncompressed only (no RLE, no 1-bit)
GIF	First frame only
PSD	Flattened (composited) image only, 8- or 16-bit

Format	Notes
TGA	
Radiance HDR	`.hdr`, RGBE; loaded as linear float — true HDR
PIC	Softimage PIC
PNM	Binary `.ppm` / `.pgm`
QOI	`.qoi` (Quite OK Image)

A `.qoi` file is recognized by its `qoif` signature (or the `.qoi` extension), so a correctly-formed file loads even if its extension is unusual.

2.1.2 DDS textures

TextureView reads both **legacy** DDS files and **modern** (DX10-header) DDS files, identified by the `DDS` file signature or the `.dds` extension. It displays **2D textures, 2D arrays, cube maps, and cube-map arrays**; 1D and volume (3D) textures are not supported.

The displayable pixel formats are listed below. Legacy FourCC codes (`DXT1`–`DXT5`, `ATI1`/`ATI2`, `BC4U`…`BC5S`, the D3D9 float codes) and legacy RGB / luminance / alpha bit-mask layouts are mapped onto their modern equivalents automatically.

Class	Formats
8-bit integer	R8, R8G8, R8G8B8A8, B8G8R8A8 (UNORM / sRGB / SNORM where applicable), A8
Packed	R10G10B10A2, B5G6R5, B5G5R5A1
16-bit	R16, R16G16, R16G16B16A16 (UNORM / SNORM / FLOAT)
32-bit float	R32, R32G32, R32G32B32A32
Block-compressed	BC1, BC2, BC3, BC4, BC5, BC6H, BC7 (sRGB variants where they exist)

Floating-point and `BC6H` formats are treated as **true HDR** and load with their full dynamic range intact, exactly like a Radiance `.hdr` file. Everything else loads as a standard low-dynamic-range image. Block-compressed formats require a GPU that supports BC texture compression; on a GPU that does not, TextureView reports this instead of showing corrupt pixels.

Chapter 3

The window and its controls

A TextureView window is intentionally minimal: a dark window whose title bar is transparent and whose title is hidden, so the image is drawn edge to edge behind it. The standard close, minimize, and zoom buttons remain in the top-left corner. A window opens at the image's pixel size, shrunk to fit when the image would otherwise be too large for the screen. When the image does not match the window's aspect ratio, the surrounding letterbox is a soft frosted blur rather than a hard black bar.

Because there is no visible title bar to grab, you move a window by **dragging the image background** itself.

All controls live in **frosted "pill" bars** floating over the image, so they stay legible on top of any content. There are two of them:

- **Top bar** (in the title bar, beside the window buttons): **Filter**, **Camera**, and the **HDR** button. This bar is always present.
- **Lower-left bar**: the **array/slice** and **mip** controls. This bar appears only when the texture actually has multiple slices or multiple mip levels — a plain 2D image with no mips shows no lower-left bar, because there is nothing there to configure.

Throughout the app, controls that do not apply to the current view are disabled rather than hidden, and a few combinations adjust each other automatically. Those interactions are noted in the relevant sections below.

Chapter 4

Navigating the image

The **Camera** segmented control (top bar) selects how the image is projected into the window. It is always available.

- **Fill** — the image is aspect-fit to the window, centered, with letterboxing as needed. This is the default.
- **Zoom** — free navigation. **Two-finger scroll** pans the image; a **pinch** gesture zooms in and out, centered on the pointer so the point beneath the cursor stays fixed. Starting a pan or pinch while in *Fill* automatically switches to *Zoom* first; because *Fill* is exactly the un-zoomed starting point, the transition is seamless and the image does not jump.
- **360°** — treats the image as an **equirectangular panorama** (an HDRI environment) and views it from inside a sphere. **Two-finger scroll** steers the look direction (yaw and pitch); **pinch** changes the field of view. This is the natural way to inspect a captured environment map or sky.

The 360° mode wraps a single image (one slice, one mip) onto the sphere, so while it is active it locks the slice- and mip-**mode** switches to *Single*. The slice and mip menus themselves remain available — you can still choose *which* slice or mip is wrapped onto the sphere — and the *Linear* and *Point* filters still apply. Only the trilinear *Mip* filter, which requires the full mip chain, is unavailable in 360°.

Chapter 5

Slices and cube faces

When a texture has more than one array slice (including the six faces of a cube map), the **lower-left bar** gains an array section with two controls.

- **Slice / Atlas** toggle:
  - **Slice** shows one slice at a time, chosen with the adjacent pop-up menu.
  - **Atlas** lays *every* slice out at once in a square grid (`ceil(√n) × ceil(√n)`), so the entire array is visible at once.
- **Slice menu** — selects the visible slice in *Slice* mode. Plain arrays are labeled `Slice 0`, `Slice 1`, …. Cube maps are labeled by face — `+X`, `-X`, `+Y`, `-Y`, `+Z`, `-Z` — and cube *arrays* are labeled `Cube 0 +X`, `Cube 0 -X`, and so on. While *Atlas* is showing every slice, this menu is disabled because there is no single "current" slice.

5.1 Picking a slice out of the Atlas

In *Atlas* mode you can select one slice directly: **hold Shift** and move the pointer over the grid. The cell under the cursor is highlighted; **Shift-click** it to return to *Slice* mode showing that slice. The top-bar and menu controls update to match.

Chapter 6

Mip levels

When a texture has a mip chain (more than one level), the **lower-left bar** gains a mip section.

- **Single / Combined** toggle:
  - **Single** shows one mip level, chosen with the mip pop-up menu.
  - **Combined** draws the *entire* mip chain at once as a montage, each level at its own native size, so you can compare levels and inspect the downsampling.
- **Mip menu** — selects the visible level in *Single* mode. Each entry shows the level index and its pixel dimensions, e.g. `0: 1024x1024`, `1: 512x512`. In *Combined* mode the menu is disabled, since all levels are shown together.

6.1 Isolating a single mip level

To isolate one level from the *Combined* montage, **hold Option** and move the pointer over a tile to highlight it, then **Option-click** it. TextureView switches to a single-slice, single-mip view of the clicked slice and level. In an Atlas + Combined view, this exits both the grid and the montage to focus on the single sub-image selected.

Chapter 7

Sampling and filtering

The **Filter** segmented control (top bar) chooses how pixels are sampled when the image is scaled. *Linear* and *Point* are always available; *Mip* appears only when the texture has a mip chain.

- **Linear** — bilinear filtering (smooth) of the currently selected mip level.
- **Point** — nearest-neighbor sampling, showing crisp individual texels. Use this to inspect the raw texel grid.
- **Mip** — trilinear filtering across the full mip chain, with the level of detail chosen automatically from the on-screen scale. This is meaningful only in single-mip *Single* view; it is unavailable in the *Combined* montage and in 360° mode, both of which draw mips at fixed sizes.

Chapter 8

Exposure and dynamic range

The **HDR** button (top bar) opens a popover with three controls governing dynamic range and brightness. These matter most for HDR sources (`.hdr` files and floating-point / `BC6H` DDS textures), but exposure works on any image.

8.1 Output: SDR / HDR

Selects whether the image is presented in standard dynamic range or true high dynamic range.

- **HDR** is available only on a display that actually has extended-dynamic-range (EDR) headroom. On a display without it, the HDR segment is disabled and a tooltip explains why.
- The *effective* output is your choice **clamped by the current display**. If you drag a window from an HDR-capable display to one without headroom, TextureView automatically falls back to SDR for that screen, and switches back when you return — your underlying preference is remembered either way.

When an HDR source opens, TextureView defaults to **HDR output on a capable display** (showing real, un-tone-mapped values) and to SDR with tone mapping otherwise. Ordinary LDR images open in SDR and are left untouched.

8.2 Exposure

A slider from **−8 to +8 EV** with a live readout (for example, `+1.5 EV`). Each stop multiplies the linear image values by a factor of two, letting you brighten deep shadows or pull down blown-out highlights to reveal the values actually stored. Exposure applies in every output and camera mode.

8.3 Tone map

A dropdown of tone-mapping operators that compress HDR values into a displayable SDR range:

None, **Reinhard**, **Reinhard-Jodie**, **ACES**, **AgX**, and **Hable**.

The tone-map operator applies **only to SDR output**. When HDR output is effectively active the dropdown is disabled, because HDR output shows the real values directly rather than a tone-

mapped approximation. For HDR sources displayed in SDR, the default operator is **AgX**; LDR images default to **None**.

Chapter 9

Saving as DDS

File > Save As DDS... (`Cmd-S`) writes the texture in the front window to a DDS file. The export preserves the texture's pixel format, full mip chain, array slices, and cube-map faces, producing a modern (DX10-header) DDS. The menu item is disabled when no image window is open.

Chapter 10

Exporting

File > Export As... (`Cmd-E`) re-encodes the **currently selected slice and mip** — at its native resolution — to a chosen image format. Unlike *Save As DDS…*, which copies the whole texture losslessly, *Export As…* renders one image through the GPU and writes it in the format and options you pick.

The save dialog carries an options area whose contents depend on the chosen **Format**:

Format	Options
QOI, BMP	—
PNG	Compression level (0–9); Filter (Auto, None, Sub, Up, Average, Paeth)
TGA	Run-length encoding
JPEG	Quality (1–100)
HDR	— (32-bit floating point, raw)
DDS	Pixel format (see below); for **BC1**/**BC3**, a Quality of Normal or High

For **DDS** you choose the stored pixel format: 8-bit and packed formats (R8, RG8, RGBA8 / BGRA8 in linear or sRGB, RGB10A2, B5G6R5, B5G5R5A1, B4G4R4A4, A4B4G4R4), floating-point formats (R/RG/RGBA 16- and 32-bit, plus RG11B10 and the shared-exponent RGB9E5), and the block-compressed formats **BC1, BC3, BC4, BC5**. Block compression runs on the CPU and can take a moment on large images; a progress window tracks it and offers **Cancel**.

10.1 Mip generation

DDS export can also generate a mip chain from the selected slice/mip. Two DDS-only controls drive it:

- **Mip filter**: how each smaller level is produced — **Linear** (box average); **Min** / **Max** / **Median** (per-2×2 reductions, useful for masks, height/AO, and denoising); or the resampling filters **Lanczos**, **Catmull-Rom**, and **Gaussian**.
- **Max LOD**: how deep the chain goes. **0** writes just the base image (no mip generation); higher values generate successive half-size levels, up to the full chain (down to 1×1).

Mips are generated on the GPU in linear light, so colours don't darken, and each level is encoded in the chosen pixel format (block-compressed levels are compressed individually).

Choosing a filter for HDR. Lanczos and **Catmull-Rom** are sharpening filters with negative side-lobes; on high-contrast HDR content (a bright highlight next to a dim background) those lobes *ring*, leaving faint dark halos around bright features. The export clamps the ringing to the local value range — so it never goes pure black or throws fireflies — but the halo can still be visible. **Linear** and **Gaussian** have no negative lobes and so never ring, which makes them the better choice for HDR; **Linear** is the default. Reach for Lanczos or Catmull-Rom when you want maximum sharpness and the source is low-contrast or standard-range (e.g. albedo or UI atlases).

10.2 Exposure and tone mapping on export

When the source is **HDR** and the chosen format is a standard-range (non-floating-point) one, the dialog shows **Exposure** and **Tone map** controls, defaulting to the current viewer settings, so the exported image matches what you see. These controls are hidden for standard-range sources and for floating-point / shared-exponent targets, which receive the raw linear values unchanged.

Chapter 11

Keyboard and gesture reference

Action	Shortcut / gesture
Open…	`Cmd-O`
Save As DDS…	`Cmd-S`
Export As…	`Cmd-E`
Close window	`Cmd-W`, `Esc`, or `Cmd-.`
Quit	`Cmd-Q`
Open a file	Drag it onto a window
Pan (Zoom mode)	Two-finger scroll
Zoom / field of view	Pinch
Target / pick an array slice	Hold **Shift** (hover, then click)
Target / pick a mip tile	Hold **Option** (hover, then click)

Starting a pan or pinch while in *Fill* automatically switches the camera to *Zoom*.

Chapter 12

Finder integration

Once TextureView is installed, two bundled system extensions add Finder previews for texture formats macOS does not handle on its own:

- **Thumbnails.** Finder shows real **icon thumbnails** for these files in icon and gallery views, instead of a generic document icon.
- **Quick Look.** Select a file and press **Space** to see a full **Quick Look preview** without opening the app.

These cover **QOI** (`.qoi`) and **PNM** (`.ppm` / `.pgm`) images — for example, press Space on a `.qoi` in Finder to preview it. DDS and Radiance `.hdr` files preview in Finder too, but on current macOS the system renders those itself.