Package: ptexr (via r-universe)

November 16, 2026

Type Package

Title Per-Face Texture Mapping for 3D Rendering

Version 2.5.4-1

Description Reads, writes, and filters per-face textures using 'Ptex' <https://github.com/wdas/ptex>. Stores textures on individual mesh faces without a global texture atlas. Provides filtering across face boundaries, shared texture caches, and a native runtime interface.

License BSD_3_clause + file LICENSE

Copyright file inst/COPYRIGHTS

URL https://github.com/tylermorganwall/ptexr

Depends R (>= 4.3.0)

LinkingTo libdeflate (>= 1.25-0)

SystemRequirements GNU make

Encoding UTF-8

NeedsCompilation yes

Roxygen list(markdown = TRUE)

Config/roxygen2/version 8.0.0

BugReports https://github.com/tylermorganwall/ptexr/issues

Config/pak/sysreqs make

Repository https://tylermorganwall.r-universe.dev

Date/Publication 2026-11-14 15:03:55 UTC

RemoteUrl https://github.com/tylermorganwall/ptexr

RemoteRef HEAD

RemoteSha 721db65a679d6dc8e8606c484d9e36070de670ae

Contents

ptex_api 2
ptex_cache 3
ptex_cache_info 3
ptex_close 4
ptex_face_info 4
ptex_info 5
ptex_open 5
ptex_sample 6
ptex_version 7
ptex_write 8

Index **10**

`ptex_api` *Acquire the Ptex Native Runtime Interface*

Description

Returns the versioned C function table for downstream packages. See `system.file("runtime-api.txt", package = "ptexr")` for integration details. The table is obtained on R's main thread; its native operations can then run on worker threads without invoking R. Keep the handle alive until all native resources and workers using it have finished. Explicitly unloading the package DLL while consumers exist is unsupported.

Usage

```
ptex_api()
```

Value

An external pointer tagged `ptex.api.v1`. It cannot be serialized.

Examples

```
api = ptex_api()
```

ptex_cache *Create a Shared Texture Cache*

Description

Memory and open-file limits are soft limits on retained cache data, not hard bounds on process memory. Concurrent lookups can temporarily exceed them. File references are returned after each lookup, so live texture handles do not prevent normal cache eviction.

Usage

```
ptex_cache(max_files = 100L, max_memory = 256 * 1024^2, premultiply = TRUE)
```

Arguments

max_files	Default 100L. Positive maximum number of cached open files.
max_memory	Default 256 * 1024^2. Cache budget in bytes; zero means unlimited. A nonzero budget is recommended for rendering.
premultiply	Default TRUE. Premultiply colour channels by alpha on input. Ptex's reduced resolutions always use premultiplied alpha.

Value

An external pointer owning a cache. Use ptex_close() to release it.

Examples

```
cache = ptex_cache(max_memory = 32 * 1024^2)
ptex_cache_info(cache)
ptex_close(cache)
```

ptex_cache_info *Inspect Cache Usage*

Description

Inspect Cache Usage

Usage

```
ptex_cache_info(cache)
```

Arguments

cache	A cache from ptex_cache().

Value

A named numeric vector of memory bytes, file counts and block reads.

`ptex_close` *Close a Texture or Cache*

Description

Close a Texture or Cache

Usage

```
ptex_close(handle)
```

Arguments

`handle`	A texture or cache handle. Closing an already closed handle is safe.

Value

Invisibly `NULL`.

`ptex_face_info` *Inspect Face Resolution and Adjacency*

Description

Inspect Face Resolution and Adjacency

Usage

```
ptex_face_info(texture, face)
```

Arguments

`texture`	A texture from `ptex_open()`.
`face`	One-based face index.

Value

A list with width, height, adjacent face and edge indices (one-based; zero means a boundary face), and the subface flag. Triangles use three edges.

ptex_info *Inspect a Ptex Texture*

Description

Inspect a Ptex Texture

Usage

```
ptex_info(texture)
```

Arguments

texture	A texture from ptex_open().

Value

A list with mesh and data types, face and channel counts, alpha channel (one-based, or zero for no alpha), and border modes.

ptex_open *Open a Per-Face Texture*

Description

Reuse one cache for a collection of textures. The file is loaded lazily; opening it does not decode every face. Textures keep their cache alive even after its R handle is explicitly closed. File contents must remain unchanged while cached; use a new cache after rewriting a file.

Usage

```
ptex_open(path, cache = ptex_cache())
```

Arguments

path	Path to a Ptex file.
cache	Default ptex_cache(). Shared cache to use for this texture.

Value

An external pointer owning a texture reference. Handles cannot be serialized; reopen files in a new R session.

Examples

```
file = tempfile(fileext = ".ptx")
ptex_write(file, list(array(c(0.1, 0.7, 0.9), c(1, 1, 3))))
texture = ptex_open(file)
ptex_info(texture)
ptex_sample(texture, face = 1, uv = c(0.5, 0.5))
ptex_close(texture)
unlink(file)
```

`ptex_sample` *Sample and Filter Per-Face Textures*

Description

Filters use the file's face adjacency to blend across seams. The footprint is a parallelogram with sides `(du1, dv1)` and `(du2, dv2)` in local face coordinates, typically derived from ray differentials. A zero footprint still permits reconstruction filtering. `filter = "point"` reads the nearest texel without blending. No gamma, sRGB or other colour conversion is applied; integer formats are normalized by Ptex to the range zero to one.

Usage

```
ptex_sample(
  texture,
  face,
  uv,
  footprint = c(0, 0, 0, 0),
  filter = "bspline",
  first_channel = 1L,
  channels = NULL,
  lerp = TRUE,
  sharpness = 0,
  no_edge_blend = FALSE
)
```

Arguments

`texture`	A texture from `ptex_open()`.
`face`	One-based face indices, one per sample, or one index for all samples.
`uv`	Two-column matrix of local `(u, v)` coordinates, or one length-two vector. Values must be in `[0, 1]`; triangle faces also require `u + v <= 1`.
`footprint`	Default `c(0, 0, 0, 0)`. Length-four vector or one matrix row per sample, ordered `du1, dv1, du2, dv2`. Components must be in `[-1, 1]`.
`filter`	Default `"bspline"`. One of `"point"`, `"bilinear"`, `"box"`, `"gaussian"`, `"bicubic"`, `"bspline"`, `"catmullrom"`, or `"mitchell"`.
`first_channel`	Default `1L`. First channel, using one-based indexing.

channels	Default NULL. Number of consecutive channels; NULL uses all channels from first_channel onward.
lerp	Default TRUE. Blend between mipmap levels.
sharpness	Default 0. General bicubic sharpness in [0, 1].
no_edge_blend	Default FALSE. Disable blending across adjacent faces.

Value

A numeric matrix with one row per sample and one column per channel.

Examples

```
# A little two-tone flag, sampled at the centres of its four texels.
file = tempfile(fileext = ".ptx")
flag = array(c(1, 0, 1, 0, 0.1, 0.7, 0.1, 0.7, 0.2, 1, 0.2, 1), c(2, 2, 3))
ptex_write(file, list(flag))
texture = ptex_open(file)
ptex_sample(texture, 1, as.matrix(expand.grid(u = c(.25, .75), v = c(.25, .75))),
            filter = "point")
ptex_close(texture)
unlink(file)
```

---

ptex_version *Ptex Library Version*

Description

Ptex Library Version

Usage

```
ptex_version()
```

Value

The bundled upstream Ptex version as a character string.

Examples

```
ptex_version()
```

ptex_write *Write Floating-Point Per-Face Textures*

Description

Writes an entire file with mipmaps. Each face is an array indexed by `[u, v, channel]` with power-of-two spatial dimensions. All faces use the same number of channels. Triangle textures use Ptex's square texel layout; sampling coordinates occupy `u >= 0, v >= 0, u + v <= 1`.

Usage

```
ptex_write(
  path,
  faces,
  mesh_type = "quad",
  adjacent_faces = NULL,
  adjacent_edges = NULL,
  alpha_channel = 0L,
  overwrite = FALSE
)
```

Arguments

`path`	Destination file path.
`faces`	Nonempty list of numeric `[u, v, channel]` arrays. Dimensions must be powers of two no larger than 32768; triangles require square arrays.
`mesh_type`	Default `"quad"`. Either `"quad"` or `"triangle"`.
`adjacent_faces`	Default `NULL`. Matrix with one row per face and four columns (three for triangles). One-based neighbouring face indices, or zero for a boundary. `NULL` makes all edges boundaries.
`adjacent_edges`	Default `NULL`. Corresponding one-based neighbour edge indices. Required when any adjacent face is nonzero.
`alpha_channel`	Default `0L`. One-based alpha channel, or zero for none.
`overwrite`	Default `FALSE`. Allow replacing an existing file. Close existing readers and create a fresh cache before reading a replacement.

Details

Adjacent face and edge matrices must describe reciprocal connections. Quad edges run counterclockwise from the bottom edge: bottom, right, top, left. Triangle edges follow Ptex's native order. This writer does not author subfaces or arbitrary metadata; existing files containing them can be read. Values are stored as floats with no colour conversion and should contain unassociated colour when an alpha channel is specified.

Value

Invisibly the normalized output path.

Examples

```
# A warm face and a cool face, joined along their shared edge.
file = tempfile(fileext = ".ptx")
faces = list(array(c(.9, .2, .05), c(1, 1, 3)),
             array(c(.05, .3, .9), c(1, 1, 3)))
neighbours = matrix(0L, 2, 4)
edges = matrix(1L, 2, 4)
neighbours[1, 2] = 2L; edges[1, 2] = 4L
neighbours[2, 4] = 1L; edges[2, 4] = 2L
ptex_write(file, faces, adjacent_faces = neighbours, adjacent_edges = edges)
texture = ptex_open(file)
ptex_sample(texture, c(1, 2), rbind(c(1, .5), c(0, .5)), footprint = c(.2, 0, 0, .2))
ptex_close(texture)
unlink(file)
```

Index

ptex_api, 2
ptex_cache, 3
ptex_cache(), *3*
ptex_cache_info, 3
ptex_close, 4
ptex_close(), *3*
ptex_face_info, 4
ptex_info, 5
ptex_open, 5
ptex_open(), *4–6*
ptex_sample, 6
ptex_version, 7
ptex_write, 8