Package 'wal'

February 2, 2024

Type Package

Title Read and Write 'wal' Bitmap Image Files and Other 'Quake' Assets

Version 0.1.1

Maintainer Tim Schäfer <ts+code@rcmd.org>

Description Read 'Quake' assets including bitmap images and textures in 'wal' file format. This package also provides support for extracting these assets from 'WAD' and 'PAK' file archives. It can also read models in 'MDL' and 'MD2' formats.

License GPL-2

Encoding UTF-8

URL https://github.com/dfsp-spirit/wal

BugReports https://github.com/dfsp-spirit/wal/issues

Imports freesurferformats (>= 0.1.12), imager, jpeg, png, spacesXYZ

Suggests knitr, rmarkdown, testthat (>= 2.1.0),

VignetteBuilder knitr

RoxygenNote 7.2.3

NeedsCompilation no

Author Tim Schäfer [aut, cre] (<https://orcid.org/0000-0002-3683-8070>)

Repository CRAN

Date/Publication 2024-02-02 18:50:02 UTC

R topics documented:

`closest.color.from.palette` *Find closest color from palette for each RGB color.*

Description

Find closest color from a palette for given colors. The similarity method used to define 'closest' is deltaE, and the input RGB colors are transformed to LAB space for the computation, assuming they are given in sRGB space.

Usage

```
closest.color.from.palette(colors_rgb, fixed_palette_rgb)
```

Arguments

`colors_rgb`	n x 3 integer matrix, the truecolor (arbitrary) input RGB colors for which you want to find the most similar colors included in the fixed palette. Range 0..255.
`fixed_palette_rgb`	the fixed palette, an n x 3 matrix of integers, representing the fixed palette colors in RGB values in range 0..255.

Value

vector of n integers, the index of the closest color into the palette for each of the `colors_rgb`.

Examples

```
colors_rgb = matrix(c(255, 0, 0, 100, 100, 100, 10, 10, 10, 5, 5, 5),
 ncol = 3, byrow = TRUE);
fixed_palette_rgb = matrix(c(255, 0, 0, 255, 5, 0, 11, 11, 11, 0, 0, 0,
 255, 255, 255), ncol = 3, byrow = TRUE);
pal_similar_colors = closest.color.from.palette(colors_rgb,
 fixed_palette_rgb);
```

`img.to.wal` *Convert image to WAL instance.*

Description

Convert an input RGB image to a WAL instance, re-mapping its colors to the WAL palette in the process and generating the mipmaps.

Usage

```
img.to.wal(in_image, apply_palette = wal::pal_q2(), wal = wal.template())
```

Arguments

`in_image`	numeric matrix with 3 dimensions: widt, height, channels. Values must be in range 0..1. This is the image format returned by `jpeg::readJPEG` and `png::readPNG`. The image can have arbitrary colors, but the colors in the final WAL image will be limited to the palette. Both the width and height must be multiples of 8. Typical idtech1/2 textures use 32, 64, ..., 512. The reason is the mipmaps.
`apply_palette`	n x 3 integer matrix, the palette for the WAL image. This is not saved to the wal image, but still required because the colors from the `in_image` will be adapted to the palette colors (replaced with the most similar ones). If the palette does not cover the colors in the source image well, the resulting WAL image will look bad (dissimilar to the source image).
`wal`	a wal instance. Note that 1 will be substracted from the data when it is written, as indices are stored 0-based in the file.

Value

wal instance

Examples

```
## Not run:
   wal = img.to.wal(jpeg::readJPEG("~/mytex.jpg"));

## End(Not run)
```

is.quakemodel *Check whether object is a Quake 1 or 2 alias model.*

Description

Check whether object is a Quake 1 or 2 alias model.

Usage

```
is.quakemodel(x)
```

Arguments

x	any R object

is.quakemodel_md2 *Check whether object is Quake 2 MD2 model*

Description

Check whether object is Quake 2 MD2 model

Usage

```
is.quakemodel_md2(x)
```

Arguments

x	any R object

is.quakemodel_mdl *Check whether object is Quake 1 MDL model*

Description

Check whether object is Quake 1 MDL model

Usage

```
is.quakemodel_mdl(x)
```

Arguments

x	any R object

pak.extract *Extract PAK contents into existing directory.*

Description

Extract PAK contents into existing directory.

Usage

```
pak.extract(pak_filepath, outdir = getwd())
```

Arguments

pak_filepath	character string, path to input PAK file.
outdir	character string, the output directory in which the files should be created. Must be writeable. The sub directories and filenames are derived from the data in the WAD.

Note

PAK files can contain a directory structure, and new subdirectories will be created under `outdir` as needed to preserve it.

pal_q1 *Get Q1 palette.*

Description

Get Q1 palette.

Usage

```
pal_q1()
```

Value

256 x 3 integer matrix, representing the RGB color values for an index into the palette.

Examples

```
pal = pal_q1();
dim(pal);
```

pal_q2 *Get Q2 palette.*

Description

Get Q2 palette.

Usage

```
pal_q2()
```

Value

256 x 3 integer matrix, representing the RGB color values for an index into the palette.

Examples

```
pal = pal_q2();
dim(pal);
```

plot.wal *S3 plot function for wal image.*

Description

S3 plot function for wal image.

Usage

```
## S3 method for class 'wal'
plot(x, ...)
```

Arguments

x	a wal instance.
...	extra args, not used.

`plotwal.mipmap` *Plot a mipmap level from a WAL image.*

Description

Plot a mipmap level from a WAL image.

Usage

```
plotwal.mipmap(wal, mip_level = 0L, apply_palette = wal::pal_q2())
```

Arguments

`wal`	a WAL image instance, as returned by `read.wal`.
`mip_level`	integer in range 0..3, the mipmap to plot. Level 0 is the original full-size image, the other ones get smaller and smaller (by factor 2 on each dimension, so 1/4th the size of their predecessor).
`apply_palette`	optional 256 x 3 integer matrix, the palette. Must contain values in range 0..255. Pass NULL if you do not want to apply any palette. The resulting `wal` object will not have an 'image' entry then.

Examples

```
## Not run:
   walf = '~/data/q2_pak0_extracted/textures/e1u2/basic1_7.wal';
   wal = read.wal(walf);
   plotwal.mipmap(wal, mip_level = 3);

## End(Not run)
```

`plotwal.rawdata` *Plot raw pixel index data as image.*

Description

Plot raw pixel index data as image.

Usage

```
plotwal.rawdata(raw_data, width, height, apply_palette = wal::pal_q2())
```

Arguments

raw_data	integer vector in containing width * height values in range 0..255, and optionally additional mipmap data at the end (which will be ignored). The raw image data. Can be a Q2 WAL data, Q1 miptex data, or anything else.
width	positive integer, the image width.
height	positive integer, the image height.
apply_palette	optional 256 x 3 integer matrix, the palette. Must contain values in range 0..255. Pass NULL if you do not want to apply any palette. The resulting wal object will not have an 'image' entry then.

Examples

```
## Not run:
# Plot the Q1 shambler skin:
mdl = read.quake.mdl("~/data/q1_pak/progs/shambler.mdl");
plotwal.rawdata(mdl$skins$skin_pic, mdl$header$skin_width,
 mdl$header$skin_height, apply_palette = pal_q1());

## End(Not run)
```

---

print.wad *S3 print function for WAD*

Description

S3 print function for WAD

Usage

```
## S3 method for class 'wad'
print(x, ...)
```

Arguments

x	wad instance
...	extra arguments, ignored

qarchive.extract *Extract any of the supported Quake archives.*

Description

Extract any of the supported Quake archives.

Usage

```
qarchive.extract(filepath, outdir, format = "auto", do_pre_checks = TRUE)
```

Arguments

filepath	character string, path to existing and readable file in PAK or WAD2 format.
outdir	character string, path to an existing and writeable output directory into which to extract the archive.
format	character string, of one 'auto' to detect from filename, 'QARCHIVE_TYPE_WAD' for WAD2, or 'QARCHIVE_TYPE_PAK' for PACK.
do_pre_checks	logical, whether to perform extra sanity checks on the other parameters.

quakemodel.to.fs.surface *Convert Quake Model to 'fs.surface' instance.*

Description

Convert Quake Model to 'fs.surface' instance.

Usage

```
quakemodel.to.fs.surface(quakemodel, frame_idx = 1L)
```

Arguments

quakemodel	an instance of quakemodel_mdl or quakemodel_md2.
frame_idx	integer, the frame to export. Quake models may contain animations made up of several frames. The mesh connectivity is unaltered between frames, but the vertex positions differ.

Value

fs.surface mesh instance, as used by the freesurferformats package.

read.pak *Read Quake PAK archive.*

Description

Read Quake PAK archive.

Usage

```
read.pak(filepath)
```

Arguments

filepath	character string, path to the file including extension.

Value

a 'pak' instance.

Examples

```
## Not run:
   pakf = '~/.steam/steam/steamapps/common/Quake/Id1/PAK0.PAK';
   pak = read.pak(pakf);

## End(Not run)
```

read.quake.md2 *Read Quake II model in MD2 format.*

Description

Read Quake II model in MD2 format.

Usage

```
read.quake.md2(filepath, anim = FALSE)
```

Arguments

filepath	character string, the path to the MD2 file
anim	logical, whether to load the whole animation (if present). Returns a list of models, the animation frames. If FALSE, only the first frame is returned.

Note

Ignore this function, it will be moved to a different package.

`read.quake.mdl` *Read Quake model in MDL format.*

Description

Read Quake model in MDL format.

Usage

```
read.quake.mdl(filepath, do_checks = FALSE)
```

Arguments

`filepath`	character string, the path to the MDL file
`do_checks`	logical, whether to perform some sanity checks on the data and warn on suspicious results.

Note

Ignore this function, it will be moved to a different package.

Examples

```
## Not run:
   mdlf = "~/data/q1_pak/progs/quaddama.mdl"
   mdl = read.quake.mdl(mdlf);

## End(Not run)
```

`read.quake1miptex` *Read a Quake mipmap texture from a WAD2 file.*

Description

Read a Quake mipmap texture from a WAD2 file.

Usage

```
read.quake1miptex(filepath, at_offset = 0L)
```

Arguments

`filepath`	character string, path to WAD file.
`at_offset`	integer, the index in the WAD file where the texture starts.

Value

a 'qmiptex' instance, its like a wall with shorter name field (16 instead of 32) and some fields (anim_name, flags, contents, value) missing.

Examples

```
## Not run:
    qm = read.quake1miptex("~/knave.wad", at_offset = 1317632);
    plotwal.mipmap(qm, apply_palette = pal_q1());

## End(Not run)
```

`read.wad` *Read Quake WAD file.*

Description

Read Quake WAD file.

Usage

```
read.wad(filepath)
```

Arguments

`filepath`	character string, path to the file.

Value

a wad instance, can be used to extract data or list contents.

Examples

```
## Not run:
   wadf = '~/knave.wad';
   wad = read.wad(wadf);
   wad.contents(wad);

## End(Not run)
```

`read.wal` *Read bitmap file in WAL format.*

Description

Read bitmap file in WAL format.

Usage

```
read.wal(filepath, hdr = TRUE, hdr_only = FALSE, apply_palette = wal::pal_q2())
```

Arguments

`filepath`	character string, path to the file including extension
`hdr`	logical, whether to return full list with header
`hdr_only`	logical, whether to read only the header
`apply_palette`	optional 256 x 3 integer matrix, the palette. Must contain values in range 0..255. Pass NULL if you do not want to apply any palette. The resulting `wal` object will not have an 'image' entry then.

Value

integer pixel matrix, each pixel value is in range 0-255 and refers to an index in a palette. The palette is NOT included in the file, so you will need to define one or get it from elsewhere to see the final image.

Examples

```
## Not run:
   walf = '~/data/q2_pak0_extracted/textures/e1u2/basic1_7.wal';
   wal = read.wal(walf);
   plot(wal);

## End(Not run)
```

`readWAL` *Read bitmap image in WAL format, returning image data only.*

Description

Read a bitmap image in WAL format, and return data in the same format as `png::readPNG` and `jpeg::readJPEG` do.

Usage

```
readWAL(filepath, apply_palette = wal::pal_q2())
```

Arguments

filepath	character string, path to the file including extension
apply_palette	optional 256 x 3 integer matrix, the palette. Must contain values in range 0..255. Pass NULL if you do not want to apply any palette. The resulting wal object will not have an 'image' entry then.

Value

numeric matrix with dimension width x height x channels, with all color values in range 0..1.

See Also

read.wal if you want to read the header and have more control.

Examples

```
## Not run:
   walf = '~/data/q2_pak0_extracted/textures/e1u2/basic1_7.wal';
   wal_image = readWAL(walf);
   dim(wal_image);

## End(Not run)
```

---

wad.contents *List WAD file contents.*

Description

List WAD file contents.

Usage

```
wad.contents(wad)
```

Arguments

wad	a wad instance, see read.wad. Alternatively a character string, which will be interpreted as a filepath to a WAD file that should be loaded.

Value

data.frame, info on the files inside the wad.

wad.extract *Extract WAD contents into existing directory.*

Description

Extract WAD contents into existing directory.

Usage

```
wad.extract(
  wad_filepath,
  outdir = getwd(),
  file_ext_mapping = wad_dir.fileext.mapping()
)
```

Arguments

wad_filepath	character string, path to input WAD file.
outdir	character string, the output directory in which the files should be created. The filenames are derived from the data in the WAD.
file_ext_mapping	named list, with keys corresponding to the type names and values are file extensions, including the dot, to use for them.

Note

One can read extracted textures with `read.quake1miptex()`.

wal.export.to.jpeg *Export wal instance to JPEG format image file.*

Description

Export wal instance to JPEG format image file.

Usage

```
wal.export.to.jpeg(wal, filepath, apply_palette = wal::pal_q2(), ...)
```

Arguments

wal	a wal instance, as returned by `read.wal`
filepath	character string, path to the JPEG file to write, including the file extension.
apply_palette	optional 256 x 3 integer matrix, the palette. Must contain values in range 0..255. Pass NULL if you do not want to apply any palette. The resulting `wal` object will not have an 'image' entry then.
...	extra parameters passed to `jpeg::writeJPEG`. Can be used to set JPEG quality.

Examples

```
## Not run:
   walf = '~/data/q2_pak0_extracted/textures/e1u2/basic1_7.wal';
   wal = read.wal(walf);
   wal.export.to.jpeg(wal, "~/basic1_7.jpg");

## End(Not run)
```

`wal.export.to.png` *Export wal instance to PNG format image file.*

Description

Export wal instance to PNG format image file.

Usage

```
wal.export.to.png(wal, filepath, apply_palette = wal::pal_q2(), ...)
```

Arguments

`wal`	a wal instance, as returned by `read.wal`
`filepath`	character string, path to the PNG file to write, including the file extension.
`apply_palette`	optional 256 x 3 integer matrix, the palette. Must contain values in range 0..255. Pass NULL if you do not want to apply any palette. The resulting `wal` object will not have an 'image' entry then.
`...`	extra parameters passed to `png::writePNG`.

Examples

```
## Not run:
   walf = '~/data/q2_pak0_extracted/textures/e1u2/basic1_7.wal';
   wal = read.wal(walf);
   wal.export.to.png(wal, "~/basic1_7.png");

## End(Not run)
```

writeWAL *Write WAL instance to bitmap file in WAL format.*

Description

Write WAL instance to bitmap file in WAL format.

Usage

```
writeWAL(filepath, wal)
```

Arguments

filepath	character string, path to the file including extension
wal	a wal instance. Note that 1 will be substracted from the data when it is written, as indices are stored 0-based in the file.

Examples

```
## Not run:
   walf = '~/data/q2_pak0_extracted/textures/e1u2/basic1_7.wal';
   wal = read.wal(walf);
   writeWAL(tempfile(fileext = ".wal"), wal);

## End(Not run)
```

Index