Package ‘piecepackr’

January 9, 2020

Encoding UTF-8
Type Package
Title Board Game Graphics
Version 1.2.1
Description Functions to make board game graphics. By default makes game diagrams, animations, and "Print & Play" lay-outs for the 'piecepack' <http://www.ludism.org/ppwiki> but can be configured to make graphics for other board game systems.
License CC BY-SA 4.0
URL https://trevorldavis.com/piecepackr
BugReports https://github.com/trevorld/piecepackr/issues
LazyLoad yes
Imports grid, grImport2, grDevices, purrr, jpeg, png, R6, stringr, tibble, tools
Suggests magick, testthat, vdiffr
SystemRequirements ghostscript
RoxygenNote 7.0.2
NeedsCompilation no
Author Trevor L Davis [aut, cre]
Maintainer Trevor L Davis <trevor.l.davis@gmail.com>
Repository CRAN
Date/Publication 2020-01-09 14:40:03 UTC

R topics documented:

  basicPieceGrob .......................... 2
game_systems ................................ 3
grid.piece .................................. 4
grid_shape_grob ................................ 8
basicPieceGrobs

Description

basicPieceGrob, pyramidTopGrob, and previewLayoutGrob are the default “grob” functions that grid.piece uses to create grid #' graphical grob objects. picturePieceGrobFn is a function that returns a “grob” function that imports graphics from files found in its directory argument.

Usage

basicPieceGrob(piece_side, suit, rank, cfg = pp_cfg())

picturePieceGrobFn(directory, filename_fn = find_pp_file)

pyramidTopGrob(piece_side, suit, rank, cfg = pp_cfg())

previewLayoutGrob(piece_side, suit, rank, cfg = pp_cfg())

Arguments

piece_side A string with piece and side separated by a underscore e.g. "coin_face"

suit Number of suit (highest rank starting from 1). The number above the total number of suits is the neutral "unsuit". and the next number above that is "no suits".

rank Number of rank (lowest rank starting from 1)

cfg Piecepack configuration list or pp_cfg object.

directory Directory that picturePieceGrobFn will look in for piece graphics.

filename_fn Function that takes arguments directory, piece_side, suit, and rank and returns the (full path) filename of the image that the function returned by picturePieceGrobFn should import.

Examples

if (require("grid")) {
  cfg <- pp_cfg(list(invert_colors=TRUE))

  pushViewport(viewport(width=unit(2, "in"), height=unit(2, "in")))
grid.draw(basicPieceGrob("tile_face", suit=1, rank=3))
popViewport()

grid.newpage()
pushViewport(viewport(width=unit(0.75, "in"), height=unit(0.75, "in")))
grid.draw(basicPieceGrob("coin_back", suit=2, rank=0, cfg=cfg))
popViewport()

grid.newpage()
pushViewport(viewport(width=unit(6, "in"), height=unit(6, "in")))
grid.draw(previewLayoutGrob("preview_layout", suit=5, rank=0, cfg=cfg))
popViewport()

grid.newpage()
pushViewport(viewport(width=unit(0.75, "in"), height=unit(0.75, "in")))
grid.draw(pyramidTopGrob("pyramid_top", suit=3, rank=5))
popViewport()

directory <- tempdir()
save_piece_images(cfg, directory=directory, format="svg", angle=0)
cfg2 <- pp_cfg(list(grob_fn=picturePieceGrobFn(directory)))

grid.newpage()
pushViewport(viewport(width=unit(0.75, "in"), height=unit(0.75, "in")))
grid.draw(pyramidTopGrob("pyramid_top", suit=3, rank=5, cfg=cfg2))
popViewport()

}
Details

Contains the following game systems:

- icehouse pieces
- piecepack plus several piecepack accessories/expansions:
  - piecepack matchsticks
  - piecepack pyramids
  - piecepack saucers
  - hexpack
  - playing cards expansion
  - dual piecepacks expansion
  - (stackpack) subpack aka mini piecepack

See Also

pp.cfg for information about the pp.cfg objects returned by game_systems.

Examples

cfgs <- game_systems()
names(cfgs)

df_tiles <- data.frame(piece_side="tile_back", x=0.5+c(3,1,3,1), y=0.5+c(3,3,1,1),
suit=NA, angle=NA, z=NA, stringsAsFactors=FALSE)
df_coins <- data.frame(piece_side="coin_back", x=rep(4:1, 4), y=rep(4:1, each=4),
suit=c(1,4,1,4,1,4,1,4,1,2,3,2,3,2,3,2),
angle=rep(c(180,0), each=8), z=1/4+1/16, stringsAsFactors=FALSE)
df <- rbind(df_tiles, df_coins)
df$cfg <- "playing_cards_expansion"
pmap_piece(df, envir=cfgs, op_scale=0.5, default.units="in")

grid.piece

Draw piecepack pieces using grid

Description

grid.piece draws a piecepack pieces onto the graphics device. pieceGrob is its grid grob counterpart. pmap_piece operates on the rows of a data frame applying pieceGrob to each row.

Usage

pmap_piece(
  .l,
  ...
  cfg = NULL,
  envir = NULL,
)
grid.piece

    trans = NULL,
    draw = TRUE,
    name = NULL,
    gp = NULL,
    vp = NULL
  )

pieceGrob(
    piece_side = "tile_back",
    suit = NA,
    rank = NA,
    cfg = pp_cfg(),
    x = unit(0.5, "npc"),
    y = unit(0.5, "npc"),
    z = NA,
    angle = 0,
    use_pictureGrob = FALSE,
    width = NA,
    height = NA,
    depth = NA,
    op_scale = 0,
    op_angle = 45,
    default.units = "npc",
    envir = NULL,
    name = NULL,
    gp = NULL,
    vp = NULL
  )

grid.piece(
    piece_side = "tile_back",
    suit = NA,
    rank = NA,
    cfg = list(),
    x = unit(0.5, "npc"),
    y = unit(0.5, "npc"),
    z = NA,
    angle = 0,
    use_pictureGrob = FALSE,
    width = NA,
    height = NA,
    depth = NA,
    op_scale = 0,
    op_angle = 45,
    default.units = "npc",
    envir = NULL,
    name = NULL,
    gp = NULL,
grid.piece

draw = TRUE,
vp = NULL
)

Arguments

.l A list of vectors, such as a data frame. The length of .l determines the number of arguments that grid.piece_wrapper will be called with. List names will be used if present.
...
Extra arguments to pass to pieceGrob.
cfg Piecepack configuration list or pp_cfg object, a list of pp_cfg objects, or a character vector of pp_cfg objects
envir Environment (or named list) containing configuration list(s).
trans Function to modify .l before drawing. Default (NULL) is to not modify .l. op_transform can help with using an oblique projection (i.e. op_scale over 0).
draw A logical value indicating whether graphics output should be produced.
name A character identifier (for grid)
gp An object of class ‘gpar’, typically the output from a call to the function ‘gpar’. This is basically a list of graphical parameter settings.
vp A grid viewport object (or NULL).
piece_side A string with piece and side separated by a underscore e.g. "coin_face"
suit Number of suit (highest rank starting from 1). The number above the total number of suits is the neutral "unsuit". and the next number above that is "no suits".
rank Number of rank (lowest rank starting from 1)
x Where to place piece on x axis of viewport
y Where to place piece on y axis of viewport
z z-coordinate of the piece. Has no effect if op_scale is 0.
angle Angle (on xy plane) to draw piece at
use_pictureGrob If TRUE instead of directly returning the grob first export to (temporary) svg and then re-import as a grImport2::pictureGrob. This is useful if drawing pieces really big or small and don’t want to play with re-configuring fontsizes.
width Width of piece
height Height of piece
depth Depth (thickness) of piece. Has no effect if op_scale is 0.
op_scale How much to scale the depth of the piece in the oblique projection (viewed from the top of the board). 0 (the default) leads to an “orthographic” projection, 0.5 is the most common scale used in the “cabinet” projection, and 1.0 is the scale used in the “cavalier” projection.
op_angle What is the angle of the oblique projection? Has no effect if op_scale is 0.
default.units A string indicating the default units to use if 'x', 'y', 'width', and/or 'height' are only given as numeric vectors.
Value

A grob object. If draw is TRUE then as a side effect will also draw it to the graphics device.

Examples

```r
if (require("grid")) {
  draw_pp_diagram <- function(cfg=pp_cfg(), op_scale=0) {
    g.p <- function(...) {
      grid.piece(..., op_scale=op_scale, cfg=cfg, default.units="in")
    }
    g.p("tile_back", x=0.5+c(3,1,3,1), y=0.5+c(3,3,1,1))
    g.p("tile_back", x=0.5+3, y=0.5+1, z=1/4+1/8)
    g.p("tile_back", x=0.5+3, y=0.5+1, z=2/4+1/8)
    g.p("die_face", suit=3, rank=5, x=1, y=1, z=1/4+1/4)
    g.p("pawn_face", x=1, y=4, z=1/4+1/2, angle=90)
    g.p("coin_back", x=3, y=4, z=1/4+1/16, angle=180)
    g.p("coin_back", suit=4, x=3, y=4, z=1/4+1/8+1/16, angle=180)
    g.p("coin_back", suit=2, x=3, y=1, z=3/4+1/8, angle=90)
  }
  # default piecepack, orthogonal projection
draw_pp_diagram(cfg=pp_cfg())
  # custom configuration, orthogonal projection
grid.newpage()
dark_colorscheme <- list(suit_color="darkred,black,darkgreen,darkblue,black",
                        invert_colors.suited=TRUE, border_color="black", border_lex=2)
traditional_ranks <- list(use_suit_as_ace=TRUE, rank_text="a,2,3,4,5")
cfg <- c(dark_colorscheme, traditional_ranks)
draw_pp_diagram(cfg=pp_cfg(cfg))
  # custom configuration, oblique projection
grid.newpage()
cfg3d <- list(width.pawn=0.75, height.pawn=0.75, depth.pawn=1,
               dm_text.pawn="", shape.pawn="convex6", invert_colors.pawn=TRUE,
               edge_color.coin="tan", edge_color.tile="tan")
cfg <- pp_cfg(c(cfg, cfg3d))
draw_pp_diagram(cfg=pp_cfg(cfg), op_scale=0.5)
  # pmap_piece lets you use data frame input
grid.newpage()
df_tiles <- data.frame(piece_side="tile_back", x=0.5+c(3,1,3,1), y=0.5+c(3,3,1,1),
                       suit=NA, angle=NA, z=NA, stringsAsFactors=FALSE)
df_coins <- data.frame(piece_side="coin_back", x=rep(4:1, 4), y=rep(4:1, each=4),
                       suit=1:16%+rep(c(1,3), each=8),
                       angle=rep(c(180,0), each=8), z=1/4+1/16, stringsAsFactors=FALSE)
df <- rbind(df_tiles, df_coins)
pmap_piece(df, cfg=cfg, op_scale=0.5, default.units="in")
}
```
grid_shape_grobs

Grid shape grob utility functions

Description

Utility functions that produce grobs of various shapes or function that returns a function that produces a grob. These are usually wrappers of polygonGrob or pathGrob.

Usage

halmaGrob(name = NULL, gp = gpar(), vp = NULL)
pyramidGrob(name = NULL, gp = gpar(), vp = NULL)
kiteGrob(name = NULL, gp = gpar(), vp = NULL)
convexGrobFn(n_vertices, t)
concaveGrobFn(n_vertices, t, r = 0.2)

Arguments

name A character identifier (for grid)
gp An object of class ‘gpar’, typically the output from a call to the function ‘gpar’. This is basically a list of graphical parameter settings.
vp A grid viewport object (or NULL).
n_vertices Number of vertices	
 t Angle (in degrees) of first vertex of shape
 r Radial distance (from 0 to 0.5)

Examples

if(require("grid")) {
  gp <- gpar(col="black", fill="yellow")
  vp <- viewport(x=1/3-1/6, width=1/3)
  grid.draw(halmaGrob(gp=gp, vp=vp))
  vp <- viewport(x=2/3-1/6, width=1/3)
  grid.draw(pyramidGrob(gp=gp, vp=vp))
  vp <- viewport(x=3/3-1/6, width=1/3)
  grid.draw(kiteGrob(gp=gp, vp=vp))
  grid.newpage()
  vp <- viewport(x=1/4, y=1/4, width=1/2, height=1/2)
  grid.draw(convexGrobFn(3, 0)(gp=gp, vp=vp))
  vp <- viewport(x=3/4, y=1/4, width=1/2, height=1/2)
  grid.draw(convexGrobFn(4, 90)(gp=gp, vp=vp))
grob_fn_helpers

vp <- viewport(x=3/4, y=3/4, width=1/2, height=1/2)
grid.draw(convexGrobFn(5, 180)(gp=gp, vp=vp))
vp <- viewport(x=1/4, y=3/4, width=1/2, height=1/2)
grid.draw(convexGrobFn(6, 270)(gp=gp, vp=vp))

grid.newpage()
vp <- viewport(x=1/4, y=1/4, width=1/2, height=1/2)
grid.draw(concaveGrobFn(3, 0, 0.1)(gp=gp, vp=vp))
vp <- viewport(x=3/4, y=1/4, width=1/2, height=1/2)
grid.draw(concaveGrobFn(4, 90, 0.2)(gp=gp, vp=vp))
vp <- viewport(x=3/4, y=3/4, width=1/2, height=1/2)
grid.draw(concaveGrobFn(5, 180, 0.3)(gp=gp, vp=vp))
vp <- viewport(x=1/4, y=3/4, width=1/2, height=1/2)
grid.draw(concaveGrobFn(6, 270)(gp=gp, vp=vp))
}

---

**grob_fn_helpers**

**pieceGrob** helper functions

### Description

gridlinesGrob returns a grob that produces gridlines. matGrob returns a grob that produces a mat.
checkersGrob returns a grob that adds checkers. hexlinesGrob returns a grob that adds hexlines.
get_shape_grob_fn returns a function that returns a grob of the piece shape. is_color_invisible tells whether the color is transparent (and hence need not be drawn).

### Usage

gridlinesGrob(col, shape = "rect", shape_t = 90, lex = 1, name = NULL)
matGrob(col, shape = "rect", shape_t = 90, mat_width = 0, name = NULL)
checkersGrob(col, shape = "rect", shape_t = 90, name = NULL)
hexlinesGrob(col, shape = "rect", name = NULL)
get_shape_grob_fn(shape, shape_t = 90, shape_r = 0.2)
is_color_invisible(col)

### Arguments

- **col** Color
- **shape** String of the shape
- **shape_t** Angle (in degrees) of first vertex of shape (ignored by many shapes).
- **lex** Multiplier to apply to the line width
name | A character identifier (for grid)
mat_width | Numeric vector of mat widths
shape_r | Radial distance (from 0 to 0.5) (ignored by most shapes)

Examples

```r
is_color_invisible("transparent")
is_color_invisible(NA)
is_color_invisible("blue")
is_color_invisible("#05AE9C")
```

```r
if (require("grid")) {
  gp <- gpar(col="black", fill="yellow")
pushViewport(viewport(x=0.25, y=0.75, width=1/2, height=1/2))
grid.draw(get_shape_grob_fn("rect")(gp=gp))
grid.draw(gridlinesGrob("blue", lex=4))
grid.draw(hexlinesGrob("green"))
popViewport()

pushViewport(viewport(x=0.75, y=0.75, width=1/2, height=1/2))
grid.draw(get_shape_grob_fn("convex6")(gp=gp))
grid.draw(checkersGrob("blue", shape="convex6"))
popViewport()

pushViewport(viewport(x=0.25, y=0.25, width=1/2, height=1/2))
grid.draw(matGrob("blue", shape="circle", mat_width=0.2))
popViewport()

pushViewport(viewport(x=0.75, y=0.25, width=1/2, height=1/2))
grid.draw(matGrob("blue", shape="rect", mat_width=c(0.2, 0.1, 0.3, 0.4)))
popViewport()
}
```

---

### op_transform

**Oblique projection helper function**

#### Description

Guesses z coordinates and sorting order to more easily make 3D graphics with pmap_piece.

#### Usage

```r
op_transform(df, ..., cfg = pp_cfg(), envir = NULL, op_angle = 45)
```
Arguments

- `df`: A data frame with coordinates and dimensions in inches
- `...`: Ignored
- `cfg`: Piecepack configuration list or `pp_cfg` object, a list of `pp_cfg` objects, or a character vector of `pp_cfg` objects
- `envir`: Environment (or named list) containing configuration list(s).
- `op_angle`: Intended oblique projection angle (used for re-sorting)

Details

The heuristics used to generate guesses for z coordinates and sorting order aren’t guaranteed to work in every case. In some cases you may get better sorting results by changing the `op_angle` or the dimensions of pieces.

Value

A tibble with extra columns added and re-sorted rows

See Also


Examples

```r
df <- tibble::tibble(piece_side="tile_back",
                    x=c(2,2,2,4,6,6,4,2,5),
                    y=c(4,4,4,4,4,2,2,2,3))
pmap_piece(df, op_angle=135, trans=op_transform,
op_scale=0.5, default.units="in")
```

Description

`get_embedded_font` returns which font is actually embedded by `cairo_pdf`. `cleave` converts a delimiter separated string into a vector. `inch(x)` is equivalent to `unit(x, "in").` `to_x`, `to_y`, `to_r`, `to_t` convert between polar coordinates (in degrees) and Cartesian coordinates.
Usage

get_embedded_font(font, char)

inch(inches)

to_x(t, r)

to_y(t, r)

to_r(x, y)

to_t(x, y)

cleave(s, sep = ",", float = FALSE, color = FALSE)

file2grob(file, distort = TRUE)

Arguments

font A character vector of font(s) passed to the fontfamily argument of grid::gpar.
char A character vector of character(s) to be embedded by grid::grid.text
inches Number representing number of inches
t Polar angle in degrees
r Radial distance
x Cartesian x coordinate
y Cartesian y coordinate
s String to convert
sep Delimiter (defaults to ",")
float If ‘TRUE’ cast to numeric
color if ‘TRUE’ convert empty strings to ‘"transparent"’
file Filename of image
distort Logical value of whether one should preserve the aspect ratio or distort to fit the area it is drawn in

Details

get_embedded_font depends on pdffonts being on the system path (on many OSes found in a poppler-utils package).

Value

get_embedded_font returns character vector of fonts that were actually embedded by cairo_pdf. NA’s means no embedded font detected: this either means that no font was found or that a color emoji font was found and instead of a font an image was embedded.
save_piece_images

Examples

to_x(90, 1)
to_y(180, 0.5)
to_t(0, -1)
to_r(0.5, 0)

cleave("0.5,0.2,0.4,0.5", float=TRUE)
cleave("black,darkred,#050EAA,,", color=TRUE)

if (require("grid")) {
  grid.rect(width=inch(1), height=inch(3), gp=gpar(fill="blue"))
}
if ((Sys.which("pdffonts") != "") && capabilities("cairo")) {
  chars <- c("a", \u2666")
  fonts <- c("sans", "Sans Noto", "Noto Sans", "Noto Sans Symbols2")
  get_embedded_font(fonts, chars)
}

save_piece_images

Save piecepack images

Description

Saves images of all individual piecepack pieces.

Usage

save_piece_images(
  cfg = pp_cfg(),
  directory = tempdir(),
  format = "svg",
  angle = 0
)

Arguments

  cfg  Piecepack configuration list
  directory  Directory where to place images
  format  Character vector of formats to save images in
  angle  Numeric vector of angles to rotate images (in degrees)

Examples

if (all(capabilities(c("cairo", "png")))) {
  cfg <- pp_cfg(list(suit_color="darkred,black,darkgreen,darkblue,green"))
  save_piece_images(cfg, directory=tempdir(), format="svg", angle=0)
save_print_and_play

save_piece_images(cfg, directory=tempdir(), format="png", angle=90)

Description

Save piecepack print-and-play (PnP) file

Usage

save_print_and_play(
  cfg = pp_cfg(),
  output_filename = "piecepack.pdf",
  size = "letter",
  pieces = c("piecepack", "matchsticks", "pyramids"),
  arrangement = "single-sided"
)

Arguments

- **cfg**: Piecepack configuration list
- **output_filename**: Filename for print-and-play file
- **size**: PnP output size (currently either "letter", "A4", or "A5")
- **pieces**: Character vector of desired PnP pieces. Supports "piecepack", "matchsticks", "pyramids", "subpack", or "all".
- **arrangement**: Either "single-sided" or "double-sided".

Examples

```r
if (capabilities("cairo")) {
  cfg <- pp_cfg(list(invert_colors.suited=TRUE))
  save_print_and_play(cfg, "my_pnp_file.pdf")
  save_print_and_play(cfg, "my_pnp_file_ds.pdf", arrangement="double-sided")
  save_print_and_play(cfg, "my_pnp_file_A4.pdf", size="A4", pieces="all")
  save_print_and_play(cfg, "my_pnp_file_A5.pdf", size="A5")
}
```
to_hexpack

Description

to_hexpack and as_pp_cfg creates piecepack configuration list R6 object. is_pp_cfg returns TRUE if object is a piecepack configuration list R6 object. as.list will convert it into a list. to_subpack and to_hexpack will attempt to generate matching (piecepack stackpack) subpack and hexpack piecepack configuration list R6 objects given a piecepack configuration.

Usage

```r
to_hexpack(cfg = pp_cfg())
to_subpack(cfg = pp_cfg())
pp_cfg(cfg = list())
is_pp_cfg(cfg)
as_pp_cfg(cfg = list())
```

Arguments

cfg List of configuration options

See Also

https://trevorldavis.com/piecepackr/configuration-lists.html for more details about piecepackr configuration lists. game_systems returns configuration list objects for several game systems.

Examples

```r
cfg <- pp_cfg(list(invert_colors=TRUE))
as.list(cfg)
is_pp_cfg(cfg)
as_pp_cfg(list(suit_color="darkred,black,darkgreen,darkblue, grey"))
cfg$get_suit_color(suit=3)
cfg$annotation_color
cfg$has_matchsticks
cfg$has_matchsticks <- TRUE
cfg$has_matchsticks
cfg$get_width("tile_back")
cfg$get_height("die_face")
cfg$get_depth("coin_face")

cfg <- list()
system.time(replicate(100, grid.piece("tile_face", 4, 4, cfg)))
```
cfg <- pp_cfg(list())

system.time(replicate(100, grid.piece("tile_face", 4, 4, cfg)))
Index

as_pp_cfg (to_hexpack), 15
basicPieceGrob (basicPieceGrobs), 2
basicPieceGrobs, 2
checkersGrob (grob_fn_helpers), 9
cleave (pp_utils), 11
concaveGrobFn (grid_shape_grobs), 8
convexGrobFn (grid_shape_grobs), 8
file2grob (pp_utils), 11
game_systems, 3, 15
get_embedded_font (pp_utils), 11
get_shape_grob_fn (grob_fn_helpers), 9
grid.piece, 4
grid_shape_grobs, 8
gridlinesGrob (grob_fn_helpers), 9
grob_fn_helpers, 9
halmaGrob (grid_shape_grobs), 8
hexlinesGrob (grob_fn_helpers), 9
inch (pp_utils), 11
is_color_invisible (grob_fn_helpers), 9
is_pp_cfg (to_hexpack), 15
kiteGrob (grid_shape_grobs), 8
matGrob (grob_fn_helpers), 9
op_transform, 10
picturePieceGrobFn (basicPieceGrobs), 2
pieceGrob (grid.piece), 4
pmap_piece (grid.piece), 4
pp_CFG, 4
pp_CFG (to_hexpack), 15
pp_utils, 11
previewLayoutGrob (basicPieceGrobs), 2
pyramidGrob (grid_shape_grobs), 8
pyramidTopGrob (basicPieceGrobs), 2
save_piece_images, 13
save_print_and_play, 14
to_hexpack, 15
to_r (pp_utils), 11
to_subpack (to_hexpack), 15
to_t (pp_utils), 11
to_x (pp_utils), 11
to_y (pp_utils), 11