Package ‘rayrender’

April 19, 2020

Type Package
Title Build and Raytrace 3D Scenes
Version 0.9.1
Maintainer Tyler Morgan-Wall <tylermw@gmail.com>
License GPL-3
Copyright file inst/COPYRIGHTS
Imports Rcpp (>= 1.0.0), parallel, assertthat, tibble, magrittr, purrr, png, raster, decido, rayimage
Suggests sf, spData
LinkingTo Rcpp, RcppThread, progress
URL https://www.rayrender.net,
      https://github.com/tylermorganwall/rayrender
RoxygenNote 7.0.2
NeedsCompilation yes
Author Tyler Morgan-Wall [aut, cph, cre]
          (https://orcid.org/0000-0002-3131-3814),
          Syoyo Fujita [ctb, cph],
          Melissa O'Neill [ctb, cph]
Repository CRAN
Date/Publication 2020-04-19 14:50:02 UTC

R topics documented:

add_object ................................................................. 2
cube ........................................................................... 3
## Description

Add Object

## Usage

```
add_object(scene, objects)
```

## Arguments

- **scene**
  - Tibble of pre-existing object locations and properties.
- **objects**
  - A tibble row or collection of rows representing each object.

## Value

Tibble of object locations and properties.
Examples

# Generate the ground and add some objects
scene = generate_ground(depth=-0.5, material = diffuse(checkercolor="blue")) %>%
  add_object(cube(x=0.7,
      material=diffuse(noise=5, noisecolor="purple", color="black", noisephase=45),
      angle=c(0, -30, 0))) %>%
  add_object(sphere(x=-0.7, radius=0.5, material=metal(color="gold")))

render_scene(scene, parallel=TRUE)

cube  Cube Object

Description
Cube Object

Usage
cube(
  x = 0,
  y = 0,
  z = 0,
  width = 1,
  xwidth = 1,
  ywidth = 1,
  zwidth = 1,
  material = diffuse(),
  angle = c(0, 0, 0),
  order_rotation = c(1, 2, 3),
  velocity = c(0, 0, 0),
  flipped = FALSE,
  scale = c(1, 1, 1)
)

Arguments

- **x**: Default '0'. x-coordinate of the center of the cube
- **y**: Default '0'. y-coordinate of the center of the cube
- **z**: Default '0'. z-coordinate of the center of the cube
- **width**: Default '1'. Cube width.
- **xwidth**: Default '1'. x-width of the cube. Overrides 'width' argument for x-axis.
- **ywidth**: Default '1'. y-width of the cube. Overrides 'width' argument for y-axis.
- **zwidth**: Default '1'. z-width of the cube. Overrides 'width' argument for z-axis.
material

Default diffuse. The material, called from one of the material functions diffuse, metal, or dielectric.

angle

Default ‘c(0, 0, 0)’. Angle of rotation around the x, y, and z axes, applied in the order specified in ‘order_rotation’.

order_rotation

Default ‘c(1, 2, 3)’. The order to apply the rotations, referring to "x", "y", and "z".

velocity

Default ‘c(0, 0, 0)’. Velocity of the cube.

flipped

Default ‘FALSE’. Whether to flip the normals.

scale

Default ‘c(1, 1, 1)’. Scale transformation in the x, y, and z directions. If this is a single value, number, the object will be scaled uniformly. Note: emissive objects may not currently function correctly when scaled.

Value

Single row of a tibble describing the cube in the scene.

Examples

#Generate a cube in the cornell box.

generate_cornell() %>%
  add_object(cube(x = 555/2, y = 100, z = 555/2,
                 xwidth = 200, ywidth = 200, zwidth = 200, angle = c(0, 30, 0))) %>%
  render_scene(lookfrom = c(278, 278, -800), lookat = c(278, 278, 0), fov = 40,
               ambient_light = FALSE, samples = 500, parallel = TRUE, clamp_value = 5)

#Generate a gold cube in the cornell box

generate_cornell() %>%
  add_object(cube(x = 555/2, y = 100, z = 555/2,
                 xwidth = 200, ywidth = 200, zwidth = 200, angle = c(0, 30, 0),
                 material = metal(color = "gold", fuzz = 0.2))) %>%
  render_scene(lookfrom = c(278, 278, -800), lookat = c(278, 278, 0), fov = 40,
               ambient_light = FALSE, samples = 500, parallel = TRUE, clamp_value = 5)

#Generate a rotated dielectric box in the cornell box

generate_cornell() %>%
  add_object(cube(x = 555/2, y = 200, z = 555/2,
                 xwidth = 200, ywidth = 100, zwidth = 200, angle = c(30, 30, 30),
                 material = dielectric())) %>%
  render_scene(lookfrom = c(278, 278, -800), lookat = c(278, 278, 0), fov = 40,
               ambient_light = FALSE, samples = 500, parallel = TRUE, clamp_value = 5)
cylinder

Cylinder Object

Description
Cylinder Object

Usage
cylinder(
  x = 0,
  y = 0,
  z = 0,
  radius = 1,
  length = 1,
  phi_min = 0,
  phi_max = 360,
  material = diffuse(),
  angle = c(0, 0, 0),
  order_rotation = c(1, 2, 3),
  velocity = c(0, 0, 0),
  flipped = FALSE,
  scale = c(1, 1, 1)
)

Arguments

- **x**: Default '0': x-coordinate of the center of the cylinder
- **y**: Default '0': y-coordinate of the center of the cylinder
- **z**: Default '0': z-coordinate of the center of the cylinder
- **radius**: Default '1': Radius of the cylinder.
- **length**: Default '1': Length of the cylinder.
- **phi_min**: Default '0': Minimum angle around the segment.
- **phi_max**: Default '360': Maximum angle around the segment.
- **material**: Default `diffuse`. The material, called from one of the material functions `diffuse`, `metal`, or `dielectric`.
- **angle**: Default `c(0, 0, 0)`: Angle of rotation around the x, y, and z axes, applied in the order specified in 'order_rotation'.
- **order_rotation**: Default `c(1, 2, 3)`: The order to apply the rotations, referring to "x", "y", and "z".
- **velocity**: Default `c(0, 0, 0)`: Velocity of the cylinder.
- **flipped**: Default 'FALSE': Whether to flip the normals.
- **scale**: Default `c(1, 1, 1)`: Scale transformation in the x, y, and z directions. If this is a single value, number, the object will be scaled uniformly. Note: emissive objects may not currently function correctly when scaled.
Value

Single row of a tibble describing the cylinder in the scene.

Examples

# Generate a cylinder in the cornell box. Add a cap to both ends.

generate_cornell() %>%
  add_object(cylinder(x = 555/2, y = 250, z = 555/2,
                      length = 300, radius = 100, material = metal())) %>%
  add_object(disk(x = 555/2, y = 400, z = 555/2,
                   radius = 100, material = metal())) %>%
  add_object(disk(x = 555/2, y = 100, z = 555/2,
                   radius = 100, material = metal(), flipped = TRUE)) %>
  render_scene(lookfrom = c(278, 278, -800) , lookat = c(278, 278, 0),
               ambient_light = FALSE, samples = 400, parallel = TRUE, clamp_value = 5)

# Rotate the cylinder

generate_cornell() %>%
  add_object(cylinder(x = 555/2, y = 250, z = 555/2,
                       length = 300, radius = 100, angle = c(0, 0, 45),
                       material = diffuse())) %>
  render_scene(lookfrom = c(278, 278, -800) , lookat = c(278, 278, 0),
               ambient_light = FALSE, samples = 400, parallel = TRUE, clamp_value = 5)

# Only render a subtended arc of the cylinder,

generate_cornell(lightintensity=3) %>%
  add_object(cylinder(x = 555/2, y = 250, z = 555/2,
                       length = 300, radius = 100, angle = c(45, 0, 0),
                       phi_min = 0, phi_max = 180,
                       material = diffuse())) %>
  render_scene(lookfrom = c(278, 278, -800) , lookat = c(278, 278, 0),
               ambient_light = FALSE, samples = 400, parallel = TRUE, clamp_value = 5)

dielectric

Dielectric (glass) Material

Description

Dielectric (glass) Material

Usage

dielectric(
  color = "white",
)
Arguments

color
Default ‘white’. The color of the surface. Can be either a hexadecimal code, R color string, or a numeric rgb vector listing three intensities between ‘0’ and ‘1’.

refraction
Default ‘1.5’. The index of refraction.

attenuation
Default ‘c(0,0,0)’. The Beer-Lambert color-channel specific exponential attenuation through the material. Higher numbers will result in less of that color making it through the material. Note: This assumes the object has a closed surface.

priority
Default ‘0’. When two dielectric materials overlap, the one with the lower priority value is used for intersection. NOTE: If the camera is placed inside a dielectric object, its priority value will not be taken into account when determining hits to other objects also inside the object.

importance_sample
Default ‘FALSE’. If ‘TRUE’, the object will be sampled explicitly during the rendering process. If the object is particularly important in contributing to the light paths in the image (e.g. light sources, refracting glass ball with caustics, metal objects concentrating light), this will help with the convergence of the image.

Value
Single row of a tibble describing the dielectric material.

Examples

```r
#Generate a checkered ground
scene = generate_ground(depth=-0.5, material = diffuse(checkercolor="grey30", checkerperiod=2))
render_scene(scene,parallel=TRUE)

#Add a glass sphere
scene %>%
  add_object(sphere(x=-0.5,radius=0.5,material=dielectric())) %>%
  render_scene(parallel=TRUE,samples=400)

#Add a rotated colored glass cube
scene %>%
  add_object(sphere(x=-0.5,radius=0.5,material=dielectric())) %>%
  add_object(cube(x=0.5,xwidth=0.5,material=dielectric(color="darkgreen"),angle=c(0,-45,0))) %>%
```
render_scene(parallel=TRUE,samples=400)

#Add an area light behind and at an angle and turn off the ambient lighting
scene %>%
  add_object(sphere(x=-0.5,radius=0.5,material=dielectric())) %>%
  add_object(cube(x=0.5,xwidth=0.5,material=dielectric(color="darkgreen"),angle=c(0,-45,0))) %>%
  add_object(yz_rect(z=-3,y=1,x=0,zwidth=3,ywidth=1.5,
  material=light(intensity=15),
  angle=c(0,-90,45), order_rotation = c(3,2,1))) %>%
  render_scene(parallel=TRUE,aperture=0, ambient_light=FALSE,samples=1000)

#Color glass using Beer-Lambert attenuation, which attenuates light on a per-channel basis as it travels through the material. This effect is what gives some types of glass a green glow at the edges. We will get this effect by setting a lower attenuation value for the 'green' (second) channel in the dielectric 'attenuation' argument.
generate_ground(depth=-0.5,material=diffuse(checkercolor="grey30",checkerperiod=2)) %>%
  add_object(sphere(z=-5,x=-0.5,y=1,material=light(intensity=10))) %>%
  add_object(cube(y=0.3,ywidth=0.1,xwidth=2,zwidth=2,
  material=dielectric(attenuation=c(1.2,0.2,1.2)),angle=c(45,110,0))) %>%
  render_scene(parallel=TRUE, samples = 1000)

#If you have overlapping dielectrics, the 'priority' value can help disambiguate what object wins. Here, I place a bubble inside a cube by setting a lower priority value and making the inner sphere have a index of refraction of 1. I also place spheres at the corners.
generate_ground(depth=-0.51,material=diffuse(checkercolor="grey30",checkerperiod=2,sigma=90)) %>%
  add_object(cube(material = dielectric(priority=2, attenuation = c(10,3,10)))) %>%
  add_object(sphere(radius=0.49,material = dielectric(priority=1, refraction=1))) %>%
  add_object(sphere(radius=0.25,x=1.01,
  material = dielectric(priority=0,refraction=1))) %>%
  add_object(sphere(radius=0.25,x=-1.01,
  material = dielectric(priority=0,attenuation = c(10,3,10)))) %>%
  render_scene(parallel=TRUE, samples = 400,lookfrom=c(5,1,5))

# We can also use this as a basic Constructive Solid Geometry interface by setting the index of refraction equal to empty space, 1. This will subtract out those regions.
# Here I make a concave lens by subtracting two spheres from a cube.
generate_ground(depth=-0.51,material=diffuse(checkercolor="grey30",checkerperiod=2,sigma=90)) %>%
  add_object(cube(material = dielectric(attenuation = c(6,6,2),priority=1))) %>%
  add_object(sphere(radius=1,x=1.01,
  material = dielectric(priority=0,refraction=1))) %>%
  add_object(sphere(radius=1,x=-1.01,
  material = dielectric(priority=0,refraction=1))) %>%
  add_object(sphere(y=10,x=3,material=light(intensit=150))) %>%
  render_scene(parallel=TRUE, samples = 400,lookfrom=c(5,3,5))
**diffuse**

**Diffuse Material**

**Description**

Diffuse Material

**Usage**

```r
diffuse(
  color = "#ffffff",
  checkercolor = NA,
  checkerperiod = 3,
  noise = 0,
  noisephase = 0,
  noiseintensity = 10,
  noisecolor = "#000000",
  gradient_color = NA,
  gradient_transpose = FALSE,
  image_texture = NA,
  alpha_texture = NA,
  fog = FALSE,
  fogdensity = 0.01,
  sigma = NULL,
  importance_sample = FALSE
)
```

**Arguments**

- **color**: Default ‘white’. The color of the surface. Can be either a hexadecimal code, R color string, or a numeric rgb vector listing three intensities between ‘0’ and ‘1’.
- **checkercolor**: Default ‘NA’. If not ‘NA’, determines the secondary color of the checkered surface. Can be either a hexadecimal code, or a numeric rgb vector listing three intensities between ‘0’ and ‘1’.
- **checkerperiod**: Default ‘3’. The period of the checker pattern. Increasing this value makes the checker pattern bigger, and decreasing it makes it smaller.
- **noise**: Default ‘0’. If not ‘0’, covers the surface in a turbulent marble pattern. This value will determine the amount of turbulence in the texture.
- **noisephase**: Default ‘0’. The phase of the noise. The noise will repeat at ‘360’.
- **noiseintensity**: Default ‘10’. Intensity of the noise.
- **noisecolor**: Default ‘#000000’. The secondary color of the noise pattern. Can be either a hexadecimal code, or a numeric rgb vector listing three intensities between ‘0’ and ‘1’.
- **gradient_color**: Default ‘NA’. If not ‘NA’, creates a secondary color for a linear gradient between the this color and color specified in ‘color’. Direction is determined by ‘gradient_transpose’.
- **image_texture**: Default ‘NA’. Creates an image texture.
- **alpha_texture**: Default ‘NA’. Creates an alpha texture.
- **fog**: Default ‘FALSE’. If TRUE, creates a fog effect.
- **fogdensity**: Default ‘0.01’. Density of the fog.
- **sigma**: Default ‘NULL’. Determines the smoothness of the fog.
- **importance_sample**: Default ‘FALSE’. If TRUE, uses importance sampling.
**gradient_transpose**
Default ‘FALSE’. If ‘TRUE’, this will use the ‘v’ coordinate texture instead of the ‘u’ coordinate texture to map the gradient.

**image_texture**
Default ‘NA’. A 3-layer RGB array or filename to be used as the texture on the surface of the object.

**alpha_texture**
Default ‘NA’. A matrix or filename (specifying a greyscale image) to be used to specify the transparency.

**fog**
Default ‘FALSE’. If ‘TRUE’, the object will be a volumetric scatterer.

**fogdensity**
Default ‘0.01’. The density of the fog. Higher values will produce more opaque objects.

**sigma**
Default ‘NULL’. A number between 0 and Infinity specifying the roughness of the surface using the Oren-Nayar microfacet model. Higher numbers indicate a roughed surface, where sigma is the standard deviation of the microfacet orientation angle. When 0, this reverts to the default lambertian behavior.

**importance_sample**
Default ‘FALSE’. If ‘TRUE’, the object will be sampled explicitly during the rendering process. If the object is particularly important in contributing to the light paths in the image (e.g. light sources, refracting glass ball with caustics, metal objects concentrating light), this will help with the convergence of the image.

**Value**
Single row of a tibble describing the diffuse material.

**Examples**

```r
#Generate the cornell box and add a single white sphere to the center
scene = generate_cornell() %>%
  add_object(sphere(x=555/2,y=555/2,z=555/2,radius=555/8,material=diffuse()))
render_scene(scene, lookfrom=c(278,278,-800), lookat = c(278,278,0), samples=500,
            aperture=0, fov=40, ambient_light=FALSE, parallel=TRUE)

#Add a checkered rectangular cube below
scene = scene %>%
  add_object(cube(x=555/2,y=555/8,z=555/2,xwidth=555/2,ywidth=555/4,zwidth=555/2,
                  material = diffuse(checkercolor="purple",checkerperiod=20)))
render_scene(scene, lookfrom=c(278,278,-800), lookat = c(278,278,0), samples=500,
            aperture=0, fov=40, ambient_light=FALSE, parallel=TRUE)

#Add a marbled sphere
scene = scene %>%
  add_object(sphere(x=555/2+555/4,y=555/2,z=555/2,radius=555/8,
                  material = diffuse(noise=1/20)))
```
```r
render_scene(scene, lookfrom=c(278,278,-800), lookat = c(278,278,0), samples=500,
aperture=0, fov=40, ambient_light=FALSE, parallel=TRUE)

# Add an orange volumetric (fog) cube
scene = scene %>%
  add_object(cube(x=555/2-555/4,y=555/2,z=555/2,xwidth=555/4,ywidth=555/4,zwidth=555/4,
  material = diffuse(fog=TRUE, fogdensity=0.05, color="orange")))

render_scene(scene, lookfrom=c(278,278,-800), lookat = c(278,278,0), samples=500,
aperture=0, fov=40, ambient_light=FALSE, parallel=TRUE)

# ' # Add an line segment with a color gradient
scene = scene %>%
  add_object(segment(start = c(555,450,450),end=c(0,450,450),radius = 50,
  material = diffuse(color="#1f7326", gradient_color = "#a60d0d")))

render_scene(scene, lookfrom=c(278,278,-800), lookat = c(278,278,0), samples=500,
aperture=0, fov=40, ambient_light=FALSE, parallel=TRUE)
```

disk

**Disk Object**

**Description**

Disk Object

**Usage**

disk(
  x = 0,
y = 0,
z = 0,
  radius = 1,
  inner_radius = 0,
  material = diffuse(),
  angle = c(0, 0, 0),
  order_rotation = c(1, 2, 3),
  velocity = c(0, 0, 0),
  flipped = FALSE,
  scale = c(1, 1, 1)
)

**Arguments**

- `x` Default '0'. x-coordinate of the center of the disk
- `y` Default '0'. y-coordinate of the center of the disk
z Default ‘0’. z-coordinate of the center of the disk.

radius Default ‘1’. Radius of the disk.

inner_radius Default ‘0’. Inner radius of the disk.

material Default diffuse. The material, called from one of the material functions diffuse, metal, or dielectric.

angle Default ‘c(0, 0, 0)’. Angle of rotation around the x, y, and z axes, applied in the order specified in ‘order_rotation’.

order_rotation Default ‘c(1, 2, 3)’. The order to apply the rotations, referring to "x", "y", and "z".

velocity Default ‘c(0, 0, 0)’. Velocity of the disk.

flipped Default ‘FALSE’. Whether to flip the normals.

scale Default ‘c(1, 1, 1)’. Scale transformation in the x, y, and z directions. If this is a single value, number, the object will be scaled uniformly. Note: emissive objects may not currently function correctly when scaled.

Value

Single row of a tibble describing the disk in the scene.

Examples

#Generate a disk in the cornell box.

generate_cornell() %>%
  add_object(disk(x = 555/2, y = 50, z = 555/2, radius = 150,
    material = diffuse(color = "orange"))) %>%
  render_scene(lookfrom = c(278, 278, -800) , lookat = c(278, 278, 0),
    ambient_light = FALSE, samples = 400, parallel = TRUE, clamp_value = 5)

#Rotate the disk.

generate_cornell() %>%
  add_object(disk(x = 555/2, y = 555/2, z = 555/2, radius = 150, angle = c(45, 0, 0),
    material = diffuse(color = "orange"))) %>%
  render_scene(lookfrom = c(278, 278, -800) , lookat = c(278, 278, 0),
    ambient_light = FALSE, samples = 400, parallel = TRUE, clamp_value = 5)

#Pass a value for the inner radius.

generate_cornell() %>%
  add_object(disk(x = 555/2, y = 555/2, z = 555/2, radius = 150, inner_radius = 75,
    angle = c(45, 0, 0), material = diffuse(color = "orange"))) %>%
  render_scene(lookfrom = c(278, 278, -800) , lookat = c(278, 278, 0),
    ambient_light = FALSE, samples = 400, parallel = TRUE, clamp_value = 5)
ellipsoid

Ellipsoid Object

Description

Note: light importance sampling for this shape is currently approximated by a sphere. This will fail for ellipsoids with large differences between axes.

Usage

```r
ellipsoid(
  x = 0,
  y = 0,
  z = 0,
  a = 1,
  b = 1,
  c = 1,
  material = diffuse(),
  angle = c(0, 0, 0),
  order_rotation = c(1, 2, 3),
  velocity = c(0, 0, 0),
  flipped = FALSE,
  scale = c(1, 1, 1)
)
```

Arguments

- **x**  
  Default ‘0’. x-coordinate of the center of the ellipsoid.

- **y**  
  Default ‘0’. y-coordinate of the center of the ellipsoid.

- **z**  
  Default ‘0’. z-coordinate of the center of the ellipsoid.

- **a**  
  Default ‘1’. Principal x-axis of the ellipsoid.

- **b**  
  Default ‘1’. Principal y-axis of the ellipsoid.

- **c**  
  Default ‘1’. Principal z-axis of the ellipsoid.

- **material**  
  Default `diffuse()`. The material, called from one of the material functions `diffuse`, `metal`, or `dielectric`.

- **angle**  
  Default `c(0, 0, 0)`. Angle of rotation around the x, y, and z axes, applied in the order specified in ‘order_rotation’.

- **order_rotation**  
  Default `c(1, 2, 3)`. The order to apply the rotations, referring to "x", "y", and "z".

- **velocity**  
  Default `c(0, 0, 0)`. Velocity of the segment.

- **flipped**  
  Default ‘FALSE’. Whether to flip the normals.

- **scale**  
  Default `c(1, 1, 1)`. Scale transformation in the x, y, and z directions. If this is a single value, number, the object will be scaled uniformly. Note: emissive objects may not currently function correctly when scaled.
Value

Single row of a tibble describing the ellipsoid in the scene.

Examples

#Generate an ellipsoid in a Cornell box

generate_cornell() %>%
  add_object(ellipsoid(x = 555/2, y = 555/2, z = 555/2,
                      a = 100, b = 50, c = 50)) %>%
  render_scene(lookfrom = c(278, 278, -800), lookat = c(278, 278, 0),
               ambient_light = FALSE, samples = 500, parallel = TRUE, clamp_value = 5)

#Change the axes to make it taller rather than wide:

generate_cornell() %>%
  add_object(ellipsoid(x = 555/2, y = 555/2, z = 555/2,
                      a = 100, b = 200, c = 100, material = metal())) %>%
  render_scene(lookfrom = c(278, 278, -800), lookat = c(278, 278, 0),
               ambient_light = FALSE, samples = 500, parallel = TRUE, clamp_value = 5)

#Rotate it and make it dielectric:

generate_cornell() %>%
  add_object(ellipsoid(x = 555/2, y = 555/2, z = 555/2,
                      a = 100, b = 200, c = 100, angle = c(0, 0, 45),
                      material = dielectric())) %>%
  render_scene(lookfrom = c(278, 278, -800), lookat = c(278, 278, 0),
               ambient_light = FALSE, samples = 500, parallel = TRUE, clamp_value = 5)

 extruded_polygon

Extruded Polygon Object

Description

Extruded Polygon Object

Usage

extruded_polygon(
  polygon = NULL,
  x = 0,
  y = 0,
  z = 0,
  plane = "xz",
  top = 1,
extruded_polygon

bottom = 0,
holes = NULL,
angle = c(0, 0, 0),
order_rotation = c(1, 2, 3),
pivot_point = c(0, 0, 0),
material = diffuse(),
center = FALSE,
flip_horizontal = FALSE,
flip_vertical = FALSE,
data_column_top = NULL,
data_column_bottom = NULL,
scale_data = 1,
scale = c(1, 1, 1),
material_id = NA

Arguments

 polygon  `sf` object or xy coordinates of polygon represented in a way that can be processed by `xy.coords()`.
x  Default `0`. x-coordinate to offset the extruded model.
y  Default `0`. y-coordinate to offset the extruded model.
z  Default `0`. z-coordinate to offset the extruded model.
plane  Default `xz`. The plane the polygon is drawn in. All possible orientations are `xz`, `zx`, `xy`, `yx`, `yz`, and `zy`.
top  Default `1`. Extruded top distance. If this equals `bottom`, the polygon will not be extruded and just the one side will be rendered.
bottom  Default `0`. Extruded bottom distance. If this equals `top`, the polygon will not be extruded and just the one side will be rendered.
holes  Default `0`. If passing in a polygon directly, this specifies which index represents the holes in the polygon. See the `earcut` function in the `decido` package for more information.
angle  Default `c(0, 0, 0)`. Angle of rotation around the x, y, and z axes, applied in the order specified in `order_rotation`.
order_rotation  Default `c(1, 2, 3)`. The order to apply the rotations, referring to "x", "y", and "z".
pivot_point  Default `c(0,0,0)`. Point at which to rotate the polygon around.
material  Default `diffuse`. The material, called from one of the material functions `diffuse`, `metal`, or `dielectric`.
center  Default `FALSE`. Whether to center the polygon at the origin.
flip_horizontal  Default `FALSE`. Flip polygon horizontally in the plane defined by `plane`.
flip_vertical  Default `FALSE`. Flip polygon vertically in the plane defined by `plane`. 
**extruded_polygon**

**data_column_top**
Default ‘NULL’. A string indicating the column in the ‘sf’ object to use to specify the top of the extruded polygon.

**data_column_bottom**
Default ‘NULL’. A string indicating the column in the ‘sf’ object to use to specify the bottom of the extruded polygon.

**scale_data**
Default ‘1’. If specifying ‘data_column_top’ or ‘data_column_bottom’, how much to scale that value when rendering.

**scale**
Default ‘c(1, 1, 1)’. Scale transformation in the x, y, and z directions. If this is a single value, number, the object will be scaled uniformly. Note: emissive objects may not currently function correctly when scaled.

**material_id**
Default ‘NA’. A unique label/number to ensure the material is shared between all triangles that make up the extruded polygon. Required if the material is ‘dielectric()’.

**Value**
Multiple row tibble describing the extruded polygon in the scene.

**Examples**

# Manually create a polygon object, here a star:

angles = seq(0,360,by=36)
xx = rev(c(rep(c(1,0.5),5),1) * sinpi(angles/180))
yy = rev(c(rep(c(1,0.5),5),1) * cospi(angles/180))
star_polygon = data.frame(x=xx,y=yy)

generate_ground(depth=0,
    material = diffuse(color="grey50",checkercolor="grey20")) %>%
add_object(extruded_polygon(star_polygon,top=0.5,bottom=0,
    material=diffuse(color="red",sigma=90))) %>%
add_object(sphere(y=4,x=-3,z=-3,material=light(intensity=30))) %>%
render_scene(parallel=TRUE,lookfrom = c(0,2,3),samples=400,lookat=c(0,0.5,0),fov=60)

# Now, let’s add a hole to the center of the polygon. We’ll make the polygon hollow by shrinking it, combining it with the normal size polygon, # and specify with the ‘hole’ argument that everything after ‘nrow(star_polygon)’ # in the following should be used to draw a hole:

hollow_star = rbind(star_polygon,0.8*star_polygon)

generate_ground(depth=-0.01,
    material = diffuse(color="grey50",checkercolor="grey20")) %>%
add_object(extruded_polygon(hollow_star,top=0.25,bottom=0, hole = nrow(star_polygon),
    material=diffuse(color="red",sigma=90))) %>%
add_object(sphere(y=4,x=-3,z=-3,material=light(intensity=30))) %>%
render_scene(parallel=TRUE,lookfrom = c(0,2,3),samples=400,lookat=c(0,0,0),fov=30)
# Render one in the y-x plane as well by changing the 'plane' argument, # as well as offset it slightly.

generate_ground(depth=-0.01, material = diffuse(color="grey50", checkercolor="grey20")) %>%
add_object(extruded_polygon(hollow_star, top=0.25, bottom=0, hole = nrow(star_polygon),
material=diffuse(color="red", sigma=90))) %>%
add_object(extruded_polygon(hollow_star, top=0.25, bottom=0, y=1.2, z=-1.2,
hole = nrow(star_polygon), plane = "yx",
material=diffuse(color="green", sigma=90))) %>%
add_object(sphere(y=4, x=-3, material=light(intensity=30))) %>%
render_scene(parallel=TRUE, lookfrom = c(0,2,4), samples=400, lookat=c(0,0.9,0), fov=40)

# Now add the zy plane:

generate_ground(depth=-0.01, material = diffuse(color="grey50", checkercolor="grey20")) %>%
add_object(extruded_polygon(hollow_star, top=0.25, bottom=0, hole = nrow(star_polygon),
material=diffuse(color="red", sigma=90))) %>%
add_object(extruded_polygon(hollow_star, top=0.25, bottom=0, y=1.2, x=1.2,
hole = nrow(star_polygon), plane = "zy",
material=diffuse(color="blue", sigma=90))) %>%
add_object(sphere(y=4, x=-3, material=light(intensity=30))) %>%
render_scene(parallel=TRUE, lookfrom = c(-4,2,4), samples=400, lookat=c(0,0.9,0), fov=40)

# We can also directly pass in sf polygons:
if("spData" %in% rownames(utils::installed.packages())) {

us_states = spData::us_states
texas = us_states[us_states$NAME == "Texas",]
# Fix no sfc class in us_states geometry data
class(texas$geometry) = c("list","sfc")
}

# This uses the raw coordinates, unless 'center = TRUE', which centers the bounding box # of the polygon at the origin.

generate_ground(depth=-0.01, material = diffuse(color="grey50", checkercolor="grey20")) %>%
add_object(extruded_polygon(texas, center = TRUE,
material=diffuse(color="#ff2222", sigma=90))) %>%
add_object(sphere(y=30, x=-30, radius=10,
material=light(color="lightblue", intensity=40))) %>%
render_scene(parallel=TRUE, lookfrom = c(0,10,-10), samples=400, lookat=c(0,0.9,0), fov=40)

# Here we use the raw coordinates, but offset the polygon manually.
`generate_ground(depth=-0.01,
    material = diffuse(color="grey50",checkercolor="grey20")) %>%
add_object(extruded_polygon(us_states, x=-96,z=-40, top=2,
    material=diffuse(color="#ff2222",sigma=90))) %>%
add_object(sphere(y=30,x=-100,radius=10,
    material=light(color="lightblue",intensity=200))) %>%
add_object(sphere(y=30,x=100,radius=10,
    material=light(color="orange",intensity=200))) %>%
render_scene(parallel=TRUE,lookfrom = c(0,120,-120),samples=400,fov=20)

#We can also set the map the height of each polygon to a column in the sf object,
#scaling it down by the maximum population state.

generate_ground(depth=0,
    material = diffuse(color="grey50",checkercolor="grey20",sigma=90)) %>%
add_object(extruded_polygon(us_states, x=-96,z=-45, data_column_top = "total_pop_15",
    scale_data = 1/max(us_states$total_pop_15)*5,
    material=diffuse(color="#ff2222",sigma=90))) %>%
add_object(sphere(y=30,x=-100,z=60,radius=10,
    material=light(color="lightblue",intensity=250))) %>%
add_object(sphere(y=30,x=100,z=-60,radius=10,
    material=light(color="orange",intensity=250))) %>%
render_scene(parallel=TRUE,lookfrom = c(-60,50,-40),lookat=c(0,-5,0),samples=400,fov=30)

generate_cornell

**Generate Cornell Box**

**Description**
Generate Cornell Box

**Usage**

```r
generate_cornell(
    light = TRUE,
    lightintensity = 5,
    lightcolor = "white",
    lightwidth = 332,
    lightdepth = 343,
    leftcolor = "#1f7326",
    rightcolor =="#a60d0d",
    roomcolor = "#bababa"
)
```
Arguments

- **light**: Default ‘TRUE’. Whether to include a light on the ceiling of the box.
- **lightintensity**: Default ‘5’. The intensity of the light.
- **lightcolor**: Default ‘white’. The color of the light.
- **lightwidth**: Default ‘332’. Width (z) of the light.
- **lightdepth**: Default ‘343’. Depth (x) of the light.
- **leftcolor**: Default ‘#1f7326’ (green).
- **rightcolor**: Default ‘#a60d0d’ (red).
- **roomcolor**: Default ‘#bababa’ (light grey).

Value

Tibble containing the scene description of the Cornell box.

Examples

```r
# Generate and render the default Cornell box.
scene = generate_cornell()
render_scene(scene, samples=400, aperture=0, fov=40, ambient_light=FALSE, parallel=TRUE)
```

```r
# Make a much smaller light in the center of the room.
scene = generate_cornell(lightwidth=200, lightdepth=200)
render_scene(scene, samples=400, aperture=0, fov=40, ambient_light=FALSE, parallel=TRUE)
```

```r
# Place a sphere in the middle of the box.
scene = scene %>%
  add_object(sphere(x=555/2, y=555/2, z=555/2, radius=555/4))
render_scene(scene, samples=400, aperture=0, fov=40, ambient_light=FALSE, parallel=TRUE)
```

```r
# Reduce "fireflies" by setting a clamp_value in render_scene()
render_scene(scene, samples=400, aperture=0, fov=40, ambient_light=FALSE, parallel=TRUE, clamp_value=3)
```

# Change the color scheme of the cornell box

```r
new_cornell = generate_cornell(leftcolor="purple", rightcolor="yellow")
render_scene(new_cornell, samples=400, aperture=0, fov=40, ambient_light=FALSE, parallel=TRUE, clamp_value=3)
```
generate_ground  

*Generate Ground*

**Description**

Generates a large sphere that can be used as the ground for a scene.

**Usage**

```r
generate_ground(
  depth = -1,
  spheresize = 1000,
  material = diffuse(color = "#ccff00")
)
```

**Arguments**

- **depth**  
  Default `-1`. Depth of the surface.

- **spheresize**  
  Default `1000`. Radius of the sphere representing the surface.

- **material**  
  Default `diffuse` with `color= "#ccff00"`. The material, called from one of the material functions `diffuse`, `metal`, or `dielectric`.

- **color**  
  Default `#ccff00`. The color of the sphere. Can be either a hexadecimal code, or a numeric rgb vector listing three intensities between `0` and `1`.

**Value**

Single row of a tibble describing the ground.

**Examples**

```r
# Generate the ground and add some objects
scene = generate_ground(depth=-0.5,
                         material = diffuse(noise=1,noisecolor="blue",noisephase=10)) %>%
           add_object(cube(x=0.7,material=diffuse(color="red"),angle=c(0,-15,0))) %>%
           add_object(sphere(x=-0.7,radius=0.5,material=dielectric(color="white")))

render_scene(scene, parallel=TRUE,lookfrom=c(0,2,10))

# Make the sphere representing the ground larger and make it a checkered surface.
scene = generate_ground(depth=-0.5, spheresize=10000,
                        material = diffuse(checkercolor="grey50")) %>%
           add_object(cube(x=0.7,material=diffuse(color="red"),angle=c(0,-15,0))) %>%
           add_object(sphere(x=-0.7,radius=0.5,material=dielectric(color="white")))

render_scene(scene, parallel=TRUE,lookfrom=c(0,1,10))
```

---

**generate_ground  

*Generate Ground*

**Description**

Generates a large sphere that can be used as the ground for a scene.

**Usage**

```r
generate_ground(
  depth = -1,
  spheresize = 1000,
  material = diffuse(color = "#ccff00")
)
```

**Arguments**

- **depth**  
  Default `-1`. Depth of the surface.

- **spheresize**  
  Default `1000`. Radius of the sphere representing the surface.

- **material**  
  Default `diffuse` with `color= "#ccff00"`. The material, called from one of the material functions `diffuse`, `metal`, or `dielectric`.

- **color**  
  Default `#ccff00`. The color of the sphere. Can be either a hexadecimal code, or a numeric rgb vector listing three intensities between `0` and `1`.

**Value**

Single row of a tibble describing the ground.

**Examples**

```r
# Generate the ground and add some objects
scene = generate_ground(depth=-0.5,
                         material = diffuse(noise=1,noisecolor="blue",noisephase=10)) %>%
           add_object(cube(x=0.7,material=diffuse(color="red"),angle=c(0,-15,0))) %>%
           add_object(sphere(x=-0.7,radius=0.5,material=dielectric(color="white")))

render_scene(scene, parallel=TRUE,lookfrom=c(0,2,10))

# Make the sphere representing the ground larger and make it a checkered surface.
scene = generate_ground(depth=-0.5, spheresize=10000,
                        material = diffuse(checkercolor="grey50")) %>%
           add_object(cube(x=0.7,material=diffuse(color="red"),angle=c(0,-15,0))) %>%
           add_object(sphere(x=-0.7,radius=0.5,material=dielectric(color="white")))

render_scene(scene, parallel=TRUE,lookfrom=c(0,1,10))
```
group_objects

Group Objects

Description

Group and transform objects together. Currently only supports a single level of grouping.

Usage

```
group_objects(
  scene,
  pivot_point = c(0, 0, 0),
  group_translate = c(0, 0, 0),
  group_angle = c(0, 0, 0),
  group_order_rotation = c(1, 2, 3),
  group_scale = c(1, 1, 1)
)
```

Arguments

- **scene**: Tibble of pre-existing object locations and properties to group together.
- **pivot_point**: Defaults to the mean location of all the objects. The point about which to pivot and move the group.
- **group_translate**: Default 'c(0,0,0)'. Vector indicating where to offset the group.
- **group_angle**: Default 'c(0,0,0)'. Angle of rotation around the x, y, and z axes, applied in the order specified in 'order_rotation'.
- **group_order_rotation**: Default 'c(1,2,3)'. The order to apply the rotations, referring to "x", "y", and "z".
- **group_scale**: Default 'c(1,1,1)'. Scaling factor for x, y, and z directions for all objects in group.

Value

Tibble of grouped object locations and properties.

Examples

```
#Generate the ground and add some objects
scene = generate_cornell() %>%
  add_object(cube(x=555/2,y=555/8,z=555/2,width=555/4)) %>%
  add_object(cube(x=555/2,y=555/4+555/16,z=555/2,width=555/8))

render_scene(scene,lookfrom=c(278,278,-800),lookat = c(278,278,0), aperture=0,
  samples=500, fov=50, parallel=TRUE, clamp_value=5)
```
#Group the entire room and rotate around its center, but keep the cubes in the same place.
scene2 = group_objects(generate_cornell(),
  pivot_point=c(555/2, 555/2, 555/2),
  group_angle=c(0, 30, 0)) %>%
  add_object(cube(x=555/2, y=555/8, z=555/2, width=555/4)) %>%
  add_object(cube(x=555/2, y=555/4 + 555/16, z=555/2, width=555/8))

render_scene(scene2, lookfrom=c(278, 278, -800), lookat = c(278, 278, 0), aperture=0, samples=500, fov=50, parallel=TRUE, clamp_value=5)

#Now group the cubes instead of the Cornell box, and rotate/translate them together
twocubes = cube(x=555/2, y=555/8, z=555/2, width=555/4) %>%
  add_object(cube(x=555/2, y=555/4 + 555/16, z=555/2, width=555/8))
scene3 = generate_cornell() %>%
  add_object(group_objects(twocubes, group_translate = c(0, 50, 0), group_angle = c(0, 45, 0)))

render_scene(scene3, lookfrom=c(278, 278, -800), lookat = c(278, 278, 0), aperture=0, samples=500, fov=50, parallel=TRUE, clamp_value=5)

#Flatten and stretch the cubes together on two axes
scene4 = generate_cornell() %>%
  add_object(group_objects(twocubes, group_translate = c(0, -40, 0),
    group_angle = c(0, 45, 0), group_scale = c(2, 0.5, 1)))

render_scene(scene4, lookfrom=c(278, 278, -800), lookat = c(278, 278, 0), aperture=0, samples=500, fov=50, parallel=TRUE, clamp_value=5)

### lambertian

**Lambertian Material (deprecated)**

**Description**
Lambertian Material (deprecated)

**Usage**
lambertian(...)

**Arguments**
...

Arguments to pass to diffuse() function.

**Value**
Single row of a tibble describing the diffuse material.
Examples

# Deprecated lambertian material. Will display a warning.

```r
scene = generate_cornell() %>%
  add_object(sphere(x=555/2,y=555/2,z=555/2, radius=555/8, material=lambertian()))
render_scene(scene, lookfrom=c(278,278,-800), lookat = c(278,278,0), samples=10,
  aperture=0, fov=40, ambient_light=FALSE, parallel=TRUE)
```

---

### light

**Light Material**

#### Description

Light Material

#### Usage

```r
light(color = "#ffffff", intensity = 10, importance_sample = TRUE)
```

#### Arguments

- **color**: Default 'white'. The color of the light. Can be either a hexadecimal code, R color string, or a numeric rgb vector listing three intensities between '0' and '1'.
- **intensity**: Default 'NA'. If a positive value, this will turn this object into a light emitting the value specified in 'color' (ignoring other properties). Higher values will produce a brighter light.
- **importance_sample**: Default 'TRUE'. Keeping this on for lights improves the convergence of the rendering algorithm, in most cases. If the object is particularly important in contributing to the light paths in the image (e.g. light sources, refracting glass ball with caustics, metal objects concentrating light), this will help with the convergence of the image.

#### Value

Single row of a tibble describing the diffuse material.

#### Examples

# Generate the cornell box without a light and add a single white sphere to the center
```r
scene = generate_cornell(light=FALSE) %>%
  add_object(sphere(x=555/2,y=555/2,z=555/2, radius=555/8, material=light()))
render_scene(scene, lookfrom=c(278,278,-800), lookat = c(278,278,0), samples=500,
  aperture=0, fov=40, ambient_light=FALSE, parallel=TRUE)
```
All gather around the orb

```r
scene = generate_ground(material = diffuse(checkercolor="grey50")) %>%
  add_object(sphere(radius=0.5,material=light(intensity=5,color="red"))) %>%
  add_object(obj_model(r_obj(), z=-3,x=-1.5,y=-1, angle=c(0,45,0))) %>%
  add_object(pig(scale=0.3, x=1.5,z=-2,y=-1.5,angle=c(0,-135,0)))
render_scene(scene, samples=500, parallel=TRUE, clamp_value=10)
```

---

**metal**

**Metallic Material**

### Description

Metallic Material

### Usage

```r
metal(
  color = "#ffffff",
  fuzz = 0,
  alpha_texture = NA,
  importance_sample = FALSE
)
```

### Arguments

- **color** Default ‘white’. The color of the sphere. Can be either a hexadecimal code, R color string, or a numeric rgb vector listing three intensities between ‘0’ and ‘1’.
- **fuzz** Default ‘0’. The roughness of the metallic surface. Maximum ‘1’.
- **alpha_texture** Default ‘NA’. A matrix or filename (specifying a greyscale image) to be used to specify the transparency.
- **importance_sample** Default ‘FALSE’. If ‘TRUE’, the object will be sampled explicitly during the rendering process. If the object is particularly important in contributing to the light paths in the image (e.g. light sources, refracting glass ball with caustics, metal objects concentrating light), this will help with the convergence of the image.

### Value

Single row of a tibble describing the metallic material.
Examples

# Generate the cornell box with a single metal sphere in the center
scene = generate_cornell() %>%
  add_object(sphere(x=555/2,y=555/2,z=555/2,radius=555/8,material=metal()))

render_scene(scene, lookfrom=c(278,278,-800), lookat = c(278,278,0), samples=500, 
aperture=0, fov=40, ambient_light=FALSE, parallel=TRUE)

# Add a rotated shiny metal cube
scene = scene %>%
  add_object(cube(x=380,y=150/2,z=200,xwidth=150,ywidth=150,zwidth=150, 
material = metal(color="#8B4513"),angle=c(0,45,0)))

render_scene(scene, lookfrom=c(278,278,-800), lookat = c(278,278,0), samples=500, 
aperture=0, fov=40, ambient_light=FALSE, parallel=TRUE)

# Add a brushed metal cube (setting the fuzz variable)
scene = scene %>%
  add_object(cube(x=150,y=150/2,z=300,xwidth=150,ywidth=150,zwidth=150, 
material = metal(color="#FAFAD2",fuzz=0.1),angle=c(0,-30,0)))

render_scene(scene, lookfrom=c(278,278,-800), lookat = c(278,278,0), samples=500, 
aperture=0, fov=40, ambient_light=FALSE, parallel=TRUE)

---

**obj** File Object

Load an obj file via a filepath. Currently only supports the diffuse texture with the ‘texture’ argument. Note: light importance sampling currently not supported for this shape.

**Usage**

```r
obj_model(
  filename,
  x = 0,
  y = 0,
  z = 0,
  scale_obj = 1,
  texture = FALSE,
  vertex_colors = FALSE,
  material = diffuse(),
  angle = c(0, 0, 0),
  order_rotation = c(1, 2, 3),
  flipped = FALSE,
  scale = c(1, 1, 1)
)
```
Arguments

filename  Filename and path to the 'obj' file. Can also be a 'txt' file, if it's in the correct 'obj' internally.
x  Default '0'. x-coordinate to offset the model.
y  Default '0'. y-coordinate to offset the model.
z  Default '0'. z-coordinate to offset the model.
scale_obj  Default '1'. Amount to scale the model. Use this to scale the object up or down on all axes, as it is more robust to numerical precision errors than the generic scale option.
texture  Default 'FALSE'. Whether to load the obj file texture.
vertex_colors  Default 'FALSE'. Set to 'TRUE' if the OBJ file has vertex colors to apply them to the model.
material  Default diffuse. The material, called from one of the material functions diffuse, metal, or dielectric.
angle  Default 'c(0, 0, 0)'. Angle of rotation around the x, y, and z axes, applied in the order specified in 'order_rotation'.
order_rotation  Default 'c(1, 2, 3)'. The order to apply the rotations, referring to "x", "y", and "z".
flipped  Default 'FALSE'. Whether to flip the normals.
scale  Default 'c(1, 1, 1)'. Scale transformation in the x, y, and z directions. If this is a single value, number, the object will be scaled uniformly. Note: emissive objects may not currently function correctly when scaled.

Value

Single row of a tibble describing the obj model in the scene.

Examples

#Load the included example R object file, by calling the r_obj() function. This returns the local file path to the 'r.txt' obj file. The file extension is "txt" due to package constraints, but the file contents are identical and it does not affect the function.

generate_ground(material = diffuse(checkercolor = "grey50")) %>%
  add_object(obj_model(y = -0.8, filename = r_obj(),
    material = metal(color = "gold", fuzz = 0.025))) %>%
  add_object(obj_model(x = 1.8, y = -0.8, filename = r_obj(),
    material = diffuse(color = "lightblue"))) %>%
  add_object(obj_model(x = -1.8, y = -0.8, filename = r_obj(),
    material = dielectric(color = "pink"))) %>%
  add_object(sphere(z = 20, x = 20, y = 20, radius = 10,
    material = light(intensity = 20))) %>%
  render_scene(parallel = TRUE, samples = 500,
    tonemap = "reinhold", aperture = 0.05, fov = 32, lookfrom = c(0, 2, 10))
# Use scale_obj to make objects bigger -- this is more robust than the generic scale argument.

generate_ground(material = diffuse(checkercolor = "grey50")) %>%
  add_object(obj_model(y = -0.8, filename = r_obj(), scale_obj = 2,
                   material = diffuse(noise = TRUE, noiseintensity = 10, noisephase=45))) %>%
  add_object(sphere(z = 20, x = 20, y = 20, radius = 10,
                   material = light(intensity = 10))) %>%
  render_scene(parallel = TRUE, samples = 500, ambient = TRUE,
               backgroundhigh="blue", backgroundlow="red",
               aperture = 0.05, fov = 32, lookfrom = c(0, 2, 10),
               lookat = c(0,1,0))

---

**pig**  
*Pig Object*

**Description**

Pig Object

**Usage**

```r
pig(
  x = 0,
  y = 0,
  z = 0,
  emotion = "neutral",
  angle = c(0, 0, 0),
  order_rotation = c(1, 2, 3),
  scale = c(1, 1, 1),
  diffuse_sigma = 0
)
```

**Arguments**

- **x**  
  Default '0'. x-coordinate of the center of the pig.

- **y**  
  Default '0'. y-coordinate of the center of the pig.

- **z**  
  Default '0'. z-coordinate of the center of the pig.

- **emotion**  
  Default 'neutral'. Other options include 'skeptical', 'worried', and 'angry'.

- **angle**  
  Default 'c(0, 0, 0)'. Angle of rotation around the x, y, and z axes, applied in the order specified in 'order_rotation'.

- **order_rotation**  
  Default 'c(1, 2, 3)'. The order to apply the rotations, referring to "x", "y", and "z".

- **scale**  
  Default 'c(1, 1, 1)'. Scale transformation in the x, y, and z directions. If this is a single value, number, the object will be scaled uniformly.
diffuse_sigma  Default '0'. Controls the Oren-Nayar sigma parameter for the pig's diffuse material.

Value

Single row of a tibble describing the pig in the scene.

Examples

# Generate a pig in the cornell box.

```
generate_cornell() %>%
  add_object(pig(x=555/2, z=555/2, y=120, scale=c(80, 80, 80), angle = c(0, 135, 0))) %>%
  render_scene(parallel=TRUE, samples=400, clamp_value=10)
```

# Show the pig staring into a mirror, worried

```
generate_cornell() %>%
  add_object(pig(x=555/2-70, z=555/2+50, y=120, scale=c(80, 80, 80),
                angle = c(0, -40, 0), emotion = "worried")) %>%
  add_object(xy_rect(x=450, z=450, y=250, ywidth=500, xwidth=200,
                     angle = c(0, 45, 0), material = metal())) %>%
  render_scene(parallel=TRUE, samples=500, clamp_value=10)
```

# Render many small pigs facing random directions, with an evil pig overlord

```
set.seed(1)
lots_of_pigs = list()

for(i in 1:10) {
  lots_of_pigs[[i]] = pig(x=50 + 450 * runif(1), z = 50 + 450 * runif(1), y=50,
                           scale = c(30, 30, 30), angle = c(0, 360*runif(1), 0), emotion = "worried")
}

many_pigs_scene = do.call(rbind, lots_of_pigs) %>%
  add_object(generate_cornell(lightintensity=20)) %>%
  add_object(pig(z=500, x=555/2, y=400, emotion = "angry",
                 scale=c(100, 100, 100), angle=c(30, 90, 0), order_rotation=c(2, 1, 3)))

render_scene(many_pigs_scene, parallel=TRUE, clamp_value=10, samples=500, tonemap = "reinhold")
```

---

**render_scene**

**Render Scene**

**Description**

Takes the scene description and renders an image, either to the device or to a filename.
render_scene

Usage

render_scene(
  scene,
  width = 400,
  height = 400,
  fov = 20,
  samples = 100,
  min_variance = 5e-05,
  min_adaptive_size = 8,
  ambient_light = FALSE,
  lookfrom = c(0, 1, 10),
  lookat = c(0, 0, 0),
  camera_up = c(0, 1, 0),
  aperture = 0.1,
  clamp_value = Inf,
  filename = NULL,
  backgroundhigh = "#80b4ff",
  backgroundlow = "#ffffff",
  shutteropen = 0,
  shutterclose = 1,
  focal_distance = NULL,
  ortho_dimensions = c(1, 1),
  tonemap = "gamma",
  bloom = TRUE,
  parallel = TRUE,
  environment_light = NULL,
  rotate_env = 0,
  debug_channel = "none",
  progress = interactive(),
  verbose = FALSE
)

Arguments

scene Tibble of object locations and properties.
width Default ‘400’. Width of the render, in pixels.
height Default ‘400’. Height of the render, in pixels.
fov Default ‘20’. Field of view, in degrees. If this is zero, the camera will use an orthographic projection. The size of the plane used to create the orthographic projection is given in argument ‘ortho_dimensions’.
samples Default ‘100’. The maximum number of samples for each pixel.
min_variance Default ‘0.00005’. Minimum acceptable variance for a block of pixels for the adaptive sampler. Smaller numbers give higher quality images, at the expense of longer rendering times. If this is set to zero, the adaptive sampler will be turned off and the renderer will use the maximum number of samples everywhere.
min_adaptive_size Default ‘8’. Width of the minimum block size in the adaptive sampler.
ambient_light: Default 'FALSE'; unless there are no emitting objects in the scene. If 'TRUE', the background will be a gradient varying from 'backgroundhigh' directly up (+y) to 'backgroundlow' directly down (-y).

lookfrom: Default 'c(0,1,10)'. Location of the camera.

lookat: Default 'c(0,0,0)'. Location where the camera is pointed.

camera_up: Default 'c(0,1,0)'. Vector indicating the "up" position of the camera.

aperture: Default '0.1'. Aperture of the camera. Smaller numbers will increase depth of field, causing less blurring in areas not in focus.

clamp_value: Default 'Inf'. If a bright light or a reflective material is in the scene, occasionally there will be bright spots that will not go away even with a large number of samples. These can be removed (at the cost of slightly darkening the image) by setting this to a small number greater than 1.

filename: Default 'NULL'. If present, the renderer will write to the filename instead of the current device.

backgroundhigh: Default '#80b4ff'. The "high" color in the background gradient. Can be either a hexadecimal code, or a numeric rgb vector listing three intensities between '0' and '1'.

backgroundlow: Default '#ffffff'. The "low" color in the background gradient. Can be either a hexadecimal code, or a numeric rgb vector listing three intensities between '0' and '1'.

shutteropen: Default '0'. Time at which the shutter is open. Only affects moving objects.

shutterclose: Default '1'. Time at which the shutter is open. Only affects moving objects.

focal_distance: Default 'NULL', automatically set to the 'lookfrom-lookat' distance unless otherwise specified.

ortho_dimensions: Default 'c(1,1)'. Width and height of the orthographic camera. Will only be used if 'fov = 0'.

tonemap: Default 'gamma'. Choose the tone mapping function. Default 'gamma' solely adjusts for gamma and clamps values greater than 1 to 1. 'reinhold' scales values by their individual color channels 'color/(1+color)' and then performs the gamma adjustment. 'uncharted' uses the mapping developed for Uncharted 2 by John Hable. 'hbd' uses an optimized formula by Jim Hejl and Richard Burgess-Dawson. Note: If set to anything other than 'gamma', objects with material 'light()' may not be anti-aliased. If 'raw', the raw array of HDR values will be returned, rather than an image or a plot.

bloom: Default 'TRUE'. Set to 'FALSE' to get the raw, pathtraced image. Otherwise, this performs a convolution of the HDR image of the scene with a sharp, long-tailed exponential kernel, which does not visibly affect dimly pixels, but does result in emitters light slightly bleeding into adjacent pixels. This provides an antialiasing effect for lights, even when tonemapping the image. Pass in a matrix to specify the convolution kernel manually, or a positive number to control the intensity of the bloom (higher number = more bloom).

parallel: Default 'FALSE'. If 'TRUE', it will use all available cores to render the image (or the number specified in 'options("cores")' if that option is not 'NULL').
environment_light
Default 'NULL'. An image to be used for the background for rays that escape the scene. Supports both HDR ('.hdr') and low-dynamic range ('.png', '.jpg') images.

rotate_env
Default '0'. The number of degrees to rotate the environment map around the scene.

debug_channel
Default 'none'. If 'depth', function will return a depth map of rays into the scene instead of an image. If 'normals', function will return an image of scene normals, mapped from 0 to 1. If 'uv', function will return an image of the uv coords. If 'variance', function will return an image showing the number of samples needed to take for each block to converge (when the

progress
Default 'TRUE' if interactive session, 'FALSE' otherwise.

verbose
Default 'FALSE'. Prints information and timing information about scene construction and raytracing progress.

Value
Raytraced plot to current device, or an image saved to a file.

Examples

#Generate a large checkered sphere as the ground
scene = generate_ground(depth=-0.5, material = diffuse(color="white", checkercolor="darkgreen"))

render_scene(scene,parallel=TRUE,samples=500)

#Add a sphere to the center
scene = scene %>%
  add_object(sphere(x=0,y=0,z=0, radius=0.5, material = diffuse(color=c(1,0,1))))

render_scene(scene,fov=20,parallel=TRUE,samples=500)

#Add a marbled cube
scene = scene %>%
  add_object(cube(x=1.1,y=0,z=0, material = diffuse(noise=3)))

render_scene(scene,fov=20,parallel=TRUE,samples=500)

#Add a metallic gold sphere
scene = scene %>%
  add_object(sphere(x=-1.1,y=0,z=0, radius=0.5, material = metal(color="gold", fuzz=0.1)))

render_scene(scene,fov=20,parallel=TRUE,samples=500)

#Lower the number of samples to render more quickly (here, we also use only one core).
render_scene(scene, samples=4)
#Add a floating R plot using the iris dataset as a png onto a floating 2D rectangle

```r
tempfileplot = tempfile()
png(filename=tempfileplot,height=400,width=800)
plot(iris$Petal.Length,iris$Sepal.Width,col=iris$Species,pch=18,cex=4)
dev.off()

image_array = aperm(png::readPNG(tempfileplot),c(2,1,3))
scene = scene %>%
  add_object(xy_rect(x=0,y=1.1,z=0,xwidth=2,angle = c(0,180,0),
    material = diffuse(image = image_array)))
render_scene(scene,fov=20,parallel=TRUE,samples=500)
```

#Move the camera

```r
render_scene(scene,lookfrom = c(7,1.5,10),lookat = c(0,0.5,0),fov=15,parallel=TRUE)
```

#Change the background gradient to a night time ambiance

```r
render_scene(scene,lookfrom = c(7,1.5,10),lookat = c(0,0.5,0),fov=15,
  backgroundhigh = "#282375", backgroundlow = "#7e77ea", parallel=TRUE,
  samples=500)
```

#Increase the aperture to blur objects that are further from the focal plane.

```r
render_scene(scene,lookfrom = c(7,1.5,10),lookat = c(0,0.5,0),fov=15,
  aperture = 0.5,parallel=TRUE,samples=500)
```

#Spin the camera around the scene, decreasing the number of samples to render faster. To make
#an animation, specify the a filename in `render_scene` for each frame and use the `av` package
#or ffmpeg to combine them all into a movie.

```r
t=1:30
xpos = 10 * sin(t*12*pi/180+pi/2)
zpos = 10 * cos(t*12*pi/180+pi/2)

#Save old par() settings
old.par = par(no.readonly = TRUE)
on.exit(par(old.par))
par(mfrow=c(5,6))
for(i in 1:30) {
  render_scene(scene, samples=5,
    lookfrom = c(xpos[i],1.5,zpos[i]),lookat = c(0,0.5,0), parallel=TRUE)
}
**r_obj**

---

**R 3D Model**

**Description**

3D obj model of the letter R, to be used with `obj_model()`.

**Usage**

`r_obj()`

**Value**

File location of the R.obj file (saved with a .txt extension)

**Examples**

```r
# Load and render the included example R object file.

generate_ground(material = diffuse(noise = TRUE, noisecolor = "grey20")) %>%
  add_object(sphere(x = 2, y = 3, z = 2, radius = 1,
    material = light(intensity = 10))) %>%
  add_object(obj_model(r_obj(), y = -1, material = diffuse(color="red"))) %>%
  render_scene(parallel=TRUE, lookfrom = c(0, 1, 10), clamp_value = 5, samples = 200)
```

---

**segment**

---

**Segment Object**

**Description**

Similar to the cylinder object, but specified by start and end points.

**Usage**

```r
segment(
  start = c(0, -1, 0),
  end = c(0, 1, 0),
  radius = 1,
  phi_min = 0,
  phi_max = 360,
  material = diffuse(),
  velocity = c(0, 0, 0),
  flipped = FALSE,
  scale = c(1, 1, 1)
)
```
Arguments

**start**
Default `c(0, -1, 0)`. Start point of the cylinder segment, specifying 'x', 'y', 'z'.

**end**
Default `c(0, 1, 0)`. End point of the cylinder segment, specifying 'x', 'y', 'z'.

**radius**
Default '1'. Radius of the segment.

**phi_min**
Default '0'. Minimum angle around the segment.

**phi_max**
Default '360'. Maximum angle around the segment.

**material**
Default `diffuse`. The material, called from one of the material functions `diffuse`, `metal`, or `dielectric`.

**velocity**
Default `c(0, 0, 0)`. Velocity of the segment.

**flipped**
Default 'FALSE'. Whether to flip the normals.

**scale**
Default `c(1, 1, 1)`. Scale transformation in the x, y, and z directions. If this is a single value, number, the object will be scaled uniformly. Notes: this will change the stated start/end position of the segment. Emissive objects may not currently function correctly when scaled.

Value

Single row of a tibble describing the segment in the scene.

Examples

```r
# Generate a segment in the cornell box.

library(scene )

generate_cornell() %>%
  add_object(segment(start = c(100, 100, 100), end = c(455, 455, 455), radius = 50)) %>%
  render_scene(lookfrom = c(278, 278, -800) ,lookat = c(278, 278, 0), fov = 40,
               ambient_light = FALSE, samples = 400, parallel = TRUE, clamp_value = 5)

# Draw a line graph representing a normal distribution, but with metal:
xvals = seq(-3, 3, length.out = 30)
yvals = dnorm(xvals)

scene_list = list()
for(i in 1:(length(xvals) - 1)) {
  scene_list[[i]] = segment(start = c(555/2 + xvals[i] * 80, yvals[i] * 800, 555/2),
              end = c(555/2 + xvals[i + 1] * 80, yvals[i + 1] * 800, 555/2),
              radius = 10,
              material = metal())
}

scene_segments = do.call(rbind,scene_list)

generate_cornell() %>%
  add_object(scene_segments) %>%
  render_scene(lookfrom = c(278, 278, -800) ,lookat = c(278, 278, 0), fov = 40,
               ambient_light = FALSE, samples = 400, parallel = TRUE, clamp_value = 5)
```
# Draw the outline of a cube:

cube_outline = segment(start = c(100, 100, 100), end = c(100, 100, 455), radius = 10) %>%
add_object(segment(start = c(100, 100, 100), end = c(100, 455, 100), radius = 10)) %>%
add_object(segment(start = c(100, 100, 455), end = c(100, 455, 455), radius = 10)) %>%
add_object(segment(start = c(100, 455, 455), end = c(455, 455, 455), radius = 10)) %>%
add_object(segment(start = c(455, 455, 100), end = c(455, 100, 100), radius = 10)) %>

generate_cornell() %>%
  add_object(cube_outline) %>%
  render_scene(lookfrom = c(278, 278, -800), lookat = c(278, 278, 0), fov = 40,
  ambient_light = FALSE, samples = 400, parallel = TRUE, clamp_value = 5)

# Shrink and rotate the cube

generate_cornell() %>%
  add_object(group_objects(cube_outline, pivot_point = c(555/2, 555/2, 555/2),
    group_angle = c(45,45,45), group_scale = c(0.5,0.5,0.5))) %>%
  render_scene(lookfrom = c(278, 278, -800), lookat = c(278, 278, 0), fov = 40,
  ambient_light = FALSE, samples = 400, parallel = TRUE, clamp_value = 5)

---

### Sphere Object

**Description**

Sphere Object

**Usage**

sphere(
  x = 0,
  y = 0,
  z = 0,
  radius = 1,
  material = diffuse(),
  angle = c(0, 0, 0),
  order_rotation = c(1, 2, 3),
  velocity = c(0, 0, 0),
)
flipped = FALSE,
scale = c(1, 1, 1)
)

Arguments

x  Default ‘0’. x-coordinate of the center of the sphere.
y  Default ‘0’. y-coordinate of the center of the sphere.
z  Default ‘0’. z-coordinate of the center of the sphere.
radius  Default ‘1’. Radius of the sphere.
material  Default diffuse. The material, called from one of the material functions diffuse, metal, or dielectric.
angle  Default ‘c(0, 0, 0)’. Angle of rotation around the x, y, and z axes, applied in the order specified in ‘order_rotation’.
order_rotation  Default ‘c(1, 2, 3)’. The order to apply the rotations, referring to "x", "y", and "z".
velocity  Default ‘c(0, 0, 0)’. Velocity of the sphere, used for motion blur.
flipped  Default ‘FALSE’. Whether to flip the normals.
scale  Default ‘c(1, 1, 1)’. Scale transformation in the x, y, and z directions. If this is a single value, number, the object will be scaled uniformly. Note: emissive objects may not currently function correctly when scaled.

Value

Single row of a tibble describing the sphere in the scene.

Examples

#Generate a sphere in the cornell box.

generate_cornell() %>%
generate_cornell()
add_object(sphere(x = 555/2, y = 555/2, z = 555/2, radius = 100)) %>%
render_scene(lookfrom = c(278, 278, -800), lookat = c(278, 278, 0), fov = 40,
ambient_light = FALSE, samples = 400, parallel = TRUE, clamp_value = 5)

#Generate a gold sphere in the cornell box

generate_cornell() %>%
generate_cornell()
add_object(sphere(x = 555/2, y = 100, z = 555/2, radius = 100,
material = metal(color = "gold", fuzz = 0.2))) %>%
render_scene(lookfrom = c(278, 278, -800), lookat = c(278, 278, 0), fov = 40,
ambient_light = FALSE, samples = 400, parallel = TRUE, clamp_value = 5)

#Add motion blur and show the sphere moving

generate_cornell() %>%
add_object(sphere(x = 555/2, y = 100, z = 555/2, radius = 100, material = metal(color = "gold", fuzz = 0.2), velocity = c(50, 0, 0)))
render_scene(lookfrom = c(278, 278, -800), lookat = c(278, 278, 0), fov = 40, ambient_light = FALSE, samples = 400, parallel = TRUE, clamp_value = 5)

triangle

Triangle Object

Description

Triangle Object

Usage

triangle(
  v1 = c(1, 0, 0),
  v2 = c(0, 1, 0),
  v3 = c(-1, 0, 0),
  n1 = rep(NA, 3),
  n2 = rep(NA, 3),
  n3 = rep(NA, 3),
  color1 = rep(NA, 3),
  color2 = rep(NA, 3),
  color3 = rep(NA, 3),
  material = diffuse()
  angle = c(0, 0, 0),
  order_rotation = c(1, 2, 3),
  flipped = FALSE,
  reversed = FALSE,
  scale = c(1, 1, 1)
)

Arguments

v1 Default 'c(1, 0, 0)'. Length-3 vector indicating the x, y, and z coordinate of the first triangle vertex.

v2 Default 'c(0, 1, 0)'. Length-3 vector indicating the x, y, and z coordinate of the second triangle vertex.

v3 Default 'c(-1, 0, 0)'. Length-3 vector indicating the x, y, and z coordinate of the third triangle vertex.

n1 Default 'NA'. Length-3 vector indicating the normal vector associated with the first triangle vertex.

n2 Default 'NA'. Length-3 vector indicating the normal vector associated with the second triangle vertex.
n3 Default ‘NA’. Length-3 vector indicating the normal vector associated with the third triangle vertex.

color1 Default ‘NA’. Length-3 vector or string indicating the color associated with the first triangle vertex. If NA but other vertices specified, color inherits from material.

color2 Default ‘NA’. Length-3 vector or string indicating the color associated with the second triangle vertex. If NA but other vertices specified, color inherits from material.

color3 Default ‘NA’. Length-3 vector or string indicating the color associated with the third triangle vertex. If NA but other vertices specified, color inherits from material.

material Default diffuse. The material, called from one of the material functions diffuse, metal, or dielectric.

angle Default ‘c(0, 0, 0)’. Angle of rotation around the x, y, and z axes, applied in the order specified in ‘order_rotation’.

order_rotation Default ‘c(1, 2, 3)’. The order to apply the rotations, referring to "x", "y", and "z".

flipped Default ‘FALSE’. Whether to flip the normals.

reversed Default ‘FALSE’. Similar to the ‘flipped’ argument, but this reverses the handedness of the triangle so it will be oriented in the opposite direction.

scale Default ‘c(1, 1, 1)’. Scale transformation in the x, y, and z directions. If this is a single value, number, the object will be scaled uniformly. Note: emissive objects may not currently function correctly when scaled.

Value

Single row of a tibble describing the XZ plane in the scene.

Examples

#Generate a triangle in the Cornell box.

```r
generate_cornell() %>%
  add_object(triangle(v1 = c(100, 100, 100), v2 = c(555/2, 455, 455), v3 = c(455, 100, 100), 
    material = diffuse(color = "purple"))) %>%
  render_scene(lookfrom = c(278, 278, -800), lookat = c(278, 278, 0), fov = 40, 
    ambient_light = FALSE, samples = 400, parallel = TRUE, clamp_value = 5)
```

#Pass individual colors to each vertex:

```r
generate_cornell() %>%
  add_object(triangle(v1 = c(100, 100, 100), v2 = c(555/2, 455, 455), v3 = c(455, 100, 100), 
    color1 = "green", color2 = "yellow", color3 = "red")) %>%
  render_scene(lookfrom = c(278, 278, -800), lookat = c(278, 278, 0), fov = 40, 
    ambient_light = FALSE, samples = 400, parallel = TRUE, clamp_value = 5)
```
xy_rect  

Rectangular XY Plane Object

Description

Rectangular XY Plane Object

Usage

```r
xy_rect(
  x = 0,
  y = 0,
  z = 0,
  xwidth = 1,
  ywidth = 1,
  material = diffuse(),
  angle = c(0, 0, 0),
  order_rotation = c(1, 2, 3),
  flipped = FALSE,
  scale = c(1, 1, 1)
)
```

Arguments

- `x`  
  Default ‘0’. x-coordinate of the center of the rectangle.

- `y`  
  Default ‘0’. x-coordinate of the center of the rectangle.

- `z`  
  Default ‘0’. z-coordinate of the center of the rectangle.

- `xwidth`  
  Default ‘1’. x-width of the rectangle.

- `ywidth`  
  Default ‘1’. y-width of the rectangle.

- `material`  
  Default `diffuse()`. The material, called from one of the material functions `diffuse`, `metal`, or `dielectric`.

- `angle`  
  Default ‘c(0, 0, 0)’. Angle of rotation around the x, y, and z axes, applied in the order specified in ‘order_rotation’.

- `order_rotation`  
  Default ‘c(1, 2, 3)’. The order to apply the rotations, referring to "x", "y", and "z".

- `flipped`  
  Default ‘FALSE’. Whether to flip the normals.

- `scale`  
  Default ‘c(1, 1, 1)’. Scale transformation in the x, y, and z directions. If this is a single value, number, the object will be scaled uniformly. Note: emissive objects may not currently function correctly when scaled.

Value

Single row of a tibble describing the XY plane in the scene.
Examples

#Generate a purple rectangle in the cornell box.

generate_cornell() %>%
    add_object(xy_rect(x = 555/2, y = 100, z = 555/2, xwidth = 200, ywidth = 200,
        material = diffuse(color = "purple"))) %>%
    render_scene(lookfrom = c(278, 278, -800), lookat = c(278, 278, 0),
        fov = 40, ambient_light = FALSE, samples = 400, parallel = TRUE, clamp_value = 5)

#Generate a gold plane in the cornell box

generate_cornell() %>%
    add_object(xy_rect(x = 555/2, y = 100, z = 555/2,
        xwidth = 200, ywidth = 200, angle = c(0, 30, 0),
        material = metal(color = "gold"))) %>%
    render_scene(lookfrom = c(278, 278, -800), lookat = c(278, 278, 0),
        fov = 40, ambient_light = FALSE, samples = 400, parallel = TRUE, clamp_value = 5)

xz_rect

Rectangular XZ Plane Object

Description

Rectangular XZ Plane Object

Usage

xz_rect(
    x = 0,
    xwidth = 1,
    z = 0,
    zwidth = 1,
    y = 0,
    material = diffuse(),
    angle = c(0, 0, 0),
    order_rotation = c(1, 2, 3),
    flipped = FALSE,
    scale = c(1, 1, 1)
)

Arguments

x
    Default ‘0’: x-coordinate of the center of the rectangle.

xwidth
    Default ‘1’: x-width of the rectangle.

z
    Default ‘0’: z-coordinate of the center of the rectangle.
### yz_rect

#### Description

Rectangular YZ Plane Object

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>zwidth</strong></td>
<td>Default ‘1’. z-width of the rectangle.</td>
<td></td>
</tr>
<tr>
<td><strong>y</strong></td>
<td>Default ‘0’. y-coordinate of the center of the rectangle.</td>
<td></td>
</tr>
<tr>
<td><strong>material</strong></td>
<td>Default <strong>diffuse</strong>. The material, called from one of the material functions <strong>diffuse</strong>, <strong>metal</strong>, or <strong>dielectric</strong>.</td>
<td></td>
</tr>
<tr>
<td><strong>angle</strong></td>
<td>Default ‘c(0, 0, 0)’. Angle of rotation around the x, y, and z axes, applied in the order specified in ‘order_rotation’.</td>
<td></td>
</tr>
<tr>
<td><strong>order_rotation</strong></td>
<td>Default ‘c(1, 2, 3)’. The order to apply the rotations, referring to &quot;x&quot;, &quot;y&quot;, and &quot;z&quot;.</td>
<td></td>
</tr>
<tr>
<td><strong>flipped</strong></td>
<td>Default ‘FALSE’. Whether to flip the normals.</td>
<td></td>
</tr>
<tr>
<td><strong>scale</strong></td>
<td>Default ‘c(1, 1, 1)’. Scale transformation in the x, y, and z directions. If this is a single value, number, the object will be scaled uniformly. Note: emissive objects may not currently function correctly when scaled.</td>
<td></td>
</tr>
</tbody>
</table>

#### Value

Single row of a tibble describing the XZ plane in the scene.

#### Examples

```r
#Generate a purple rectangle in the cornell box.
generate_cornell() %>%
  add_object(xz_rect(x = 555/2, y = 100, z = 555/2, xwidth = 200, zwidth = 200,
                   material = diffuse(color = "purple"))) %>%
  render_scene(lookfrom = c(278, 278, -800) ,lookat = c(278, 278, 0), fov = 40,
               ambient_light = FALSE, samples = 400, parallel = TRUE, clamp_value = 5)

#Generate a gold plane in the cornell box
generate_cornell() %>%
  add_object(xz_rect(x = 555/2, y = 100, z = 555/2,
                   xwidth = 200, zwidth = 200, angle = c(0, 30, 0),
                   material = metal(color = "gold"))) %>%
  render_scene(lookfrom = c(278, 278, -800) ,lookat = c(278, 278, 0), fov = 40,
               ambient_light = FALSE, samples = 400, parallel = TRUE, clamp_value = 5)
```
Usage

```r
yz_rect(
  x = 0,
  y = 0,
  z = 0,
  ywidth = 1,
  zwidth = 1,
  material = diffuse(),
  angle = c(0, 0, 0),
  order_rotation = c(1, 2, 3),
  flipped = FALSE,
  scale = c(1, 1, 1)
)
```

Arguments

- **x**: Default '0'. x-coordinate of the center of the rectangle.
- **y**: Default '0'. y-coordinate of the center of the rectangle.
- **z**: Default '0'. z-coordinate of the center of the rectangle.
- **ywidth**: Default '1'. y-width of the rectangle.
- **zwidth**: Default '1'. z-width of the rectangle.
- **material**: Default `diffuse`. The material, called from one of the material functions `diffuse`, `metal`, or `dielectric`.
- **angle**: Default `c(0, 0, 0)`. Angle of rotation around the x, y, and z axes, applied in the order specified in 'order_rotation'.
- **order_rotation**: Default `c(1, 2, 3)`. The order to apply the rotations, referring to "x", "y", and "z".
- **flipped**: Default 'FALSE'. Whether to flip the normals.
- **scale**: Default 'c(1, 1, 1)'. Scale transformation in the x, y, and z directions. If this is a single value, number, the object will be scaled uniformly. Note: emissive objects may not currently function correctly when scaled.

Value

Single row of a tibble describing the YZ plane in the scene.

Examples

```r
#Generate a purple rectangle in the cornell box.

generate_cornell() %>%
  add_object(yz_rect(x = 100, y = 100, z = 555/2, ywidth = 200, zwidth = 200,
                     material = diffuse(color = "purple"))) %>%
  render_scene(lookfrom = c(278, 278, -800), lookat = c(278, 278, 0),
               ambient_light = FALSE, samples = 400, parallel = TRUE, clamp_value = 5)

#Generate a gold plane in the cornell box
```
\begin{verbatim}
generate_cornell() %>%
  add_object(yz_rect(x = 100, y = 100, z = 555/2,
                 ywidth = 200, zwidth = 200, angle = c(0, 30, 0),
                 material = metal(color = "gold"))) %>%
  render_scene(lookfrom = c(278, 278, -800), lookat = c(278, 278, 0), fov = 40,
               ambient_light = FALSE, samples = 400, parallel = TRUE, clamp_value = 5)
\end{verbatim}
Index

add_object, 2

cube, 3
cylinder, 5
dielectric, 4, 5, 6, 12, 13, 15, 20, 26, 34, 36, 38, 39, 41, 42
diffuse, 4, 5, 9, 12, 13, 15, 20, 26, 34, 36, 38, 39, 41, 42
disk, 11
ellipsoid, 13
extruded_polygon, 14
generate_cornell, 18
generate_ground, 20
group_objects, 21
lambertian, 22
light, 23
metal, 4, 5, 12, 13, 15, 20, 24, 26, 34, 36, 38, 39, 41, 42
obj_model, 25
pig, 27
r_obj, 33
render_scene, 28
segment, 33
sphere, 35
triangle, 37
xy_rect, 39
xz_rect, 40
yz_rect, 41